DOCTORAL THESIS

Using Local Intelligence to Actively Monitor Distributed Data Sources in Decision Making Architectures

Chavez-Mora, Emma

Award date:
2013

Using local intelligence to actively monitor distributed data sources in decision making architectures

by

Emma Alejandra Chávez- Mora

BSc(Universidad Católica de la Ssma. Concepción), BEng(CompSc)(Universidad Católica de la Ssma. Concepción), MBusMgt(IT)(Universidad del Bío Bío), MIT(Hons)(Bond university)

A dissertation submitted in fulfilment of the requirements of the degree of Doctor of Philosophy in the School of Information Technology, Bond University

Supervisor

Dr. Gavin Finnie

01/09/2012

Typeset in LaTeX2e

Statement of Originality

The material in this thesis has not been previously submitted for a degree or diploma in any other university. To the best of my knowledge and belief this thesis, original, contains no material previously published or written either in whole or in part by another person except where due acknowledgement is made in the thesis itself.

Emma Alejandra Chávez - Mora

Date: September 2012

Submitted for examination: September 2012

Abstract

In data warehouse environments operational processes are used to move data from a variety of sources to a central repository usually called the warehouse. These processes include data export, data preparation, and data loading which are usually performed by using Extraction, Transformation and Loading (ETL) tools.

Most Business Intelligence (BI) applications have to consolidate data first before performing any sort of data analysis. Therefore, they use centralized decision support architectures in which commonly a data warehouse technology is employed as the way to consolidate, analyse and report data to support decision making based on all the information that has been collected from a variety of data sources.

ETL functions have been described as a critical component of a BI solution as they are responsible for retrieving data from different sources (one or more), preparing it by normalizing and transforming it in some way to then be inserted into some other repository, usually called a data warehouse, for analysis and reporting [152][56].

The main characteristics of the ETL mechanisms used in BI include the use of incremental loading techniques. This movement of data, commonly called the data refreshment, is usually driven from the central repository, which is programmed to request "query" data from the sources in a timely manner. Consequently, data refreshments are time dependent.

Research conducted in real time and active data warehousing technologies for BI have deal with near right time by mainly focusing on the response time from the "machine" point of view to capture information, rather than on improving the speed to analyse data and therefore, advise in the right time to decision makers that "something relevant" is going on. Even with faster refreshment, using streaming techniques, data analysis takes part of the last

stage after the consolidation has been performed, consequently reporting and alerts are the last activities after some sort of ETL technique has been applied. Frameworks that support real time data analysis and reporting by detecting adverse events have not yet been proposed.

Some of other research gaps identified in this area are related to how to distribute the Extraction, Transformation and Loading processes to best meet the needs of real time decision making. Traditionally the data warehouse is responsible for querying the sources in which a vast amount of data is pulled (extracted) from the sources of data to be analysed and included in reports. If included, data errors and unusual behaviours in the data extracted are discovered only after data is loaded in the data warehouse. Usually, data errors and inconsistencies depend on the mechanism programmed by the data base administrator who is mainly responsible for the performance of the architecture deployed. Thus, it might be possible to re-distribute/re-order the way traditional architectures perform data extraction/consolidation/analysis from a variety of data sources to advise decision makers in real time that unusual events have been detected.

Moreover, traditional data warehouse approaches are mainly built on static architectures that are not capable of supporting changes in their content and structure. After the literature review was conducted it was also found that the idea of dynamic adaptation to business requirements, logical adaptation rather than structural adaptation, of a DW architecture has not yet been proposed. Consequently, the demand for new BI solutions and frameworks is continuously growing but information systems research in the field has been shown to be limited [111].

To address some of the above mentioned research gaps such as the need to react in real time to unusual behaviours, this research study focuses on the area of data management and creates a framework to monitor data from a variety of data sources in real time using data warehouses. By enabling the use of local intelligence at the sources level data is monitored and pre-analysed first, if unusual behaviour in the data captured are found actions are then taken. Actions might include alerting the decision makers as unusual behaviours have been detected, doing nothing or pushing data to the warehouse as more analysis is needed. Thus, as soon as valuable data arrives at a source is analysed and only if needed data is pushed to

a central repository, a data warehouse, because more information is needed.

Given this research proposes a new way to design a monitoring infrastructure that empowers the use of local knowledge and distributes the data analysis in architectures that uses a data warehouse, the development of a prototype was needed to demonstrate the validity of the design. The framework proposed has been empirically evaluated and validated by designing and testing an architecture to be used in the health care arena to monitor unusual symptoms in coronary heart disease patients. The validation included a review of the prototype findings with specialists (decision makers), and an evaluation in terms of computational efficiency in which a part of the architecture proposed was compared and contrasted with the deployment of a traditional loading mechanism to feed a data warehouse.

The findings of this research show that by monitoring *key behaviours* in the data at the source level, evaluating them and then responding only to data abnormalities, the proposed architecture is able to respond and/or alert to unusual scenarios in a more effective and efficient way than traditional right time data warehousing strategies.

Traditional ETL mechanisms were not used, the extraction, consolidation and analysis processes have been re-distributed instead. The process of data loading is driven by the sources who are responsible for capturing and pre-analysing data changes and sending the relevant findings to the data warehouse. Thus, data is captured and pre-analysed first and the number of steps to perform data analysis for reporting and error detection have been reduced and moved to an early stage.

A form of intelligence and adaptation has been achieved by using software agents who autonomously perform the core activities of *"Transformation, Transference and Loading "* of the framework discussed and presented in this dissertation.

Acknowledgements

This dissertation represent a transition in my life in which many hours of thoughts, writing and discussions have teach me a different way to confront life. Even though the process was at time painful, it represents one of the most relevant episodes in my life so far.

There are many people to thank to whom I owe my deepest gratitude:

Above all I would like to thank my family; Pedro and Sebastian whose love, support, and encouragement at all times taught me so much about sacrifice. My mother and sisters whose words and unconditional support were crucial to finish this odyssey.

It is with immense gratitude that I acknowledge the support and help of my supervisor Professor Gavin Finnie, whose knowledge and integrity I will always respect. I could not find a better guide who was always available, patient, and wizard to answer my questions. I was fortuned to have worked with him. Thanks for your knowledge, guidance and continuous encouragement.

I would like to acknowledge the Minister of Education and the Universidad Católica de la Ssma. Concepción in Chile, whose financial support trough the Mecesup and Postgraduate scholarships helped me to maintain financially over the past years. To Bond university, in particular the IT school, to all its academic, specially Professor Paddy Krishna for his discussions and valuable feedback and the administrative staff who provided me with the space, facilities and assistance in many opportunities.

Finally yet importantly, in my daily work I have been blessed with a friendly and cheerful group of fellow students and friends. I am indebted to my many friends who supported me; Abdullah, Sabina, Bjoern, Mena, Percy, Donald, Tina, Kim and Valentina to just name a few where part of my family here in Australia.

Publications

Publications arising from this thesis

1. ©Springer Verlag.

   Chávez, E. and Finnie, G. A real time multi-agent information systems architecture for actively monitoring chronic diseases. 19th International conference on Information Systems Development, ISD 2010.

2. ©IEEE.

   Chávez, E. and Finnie, G. Empowering data sources to manage clinical data. The 23RD IEEE international symposium on computer-based medical systems, CBMS 2010.

3. ©IEEE.

   Chávez, E. and Finnie, G. A push-based health information framework to monitor heart disease patients in real time. 2011 IEEE International Conference on Medical Information and Bioengineering, ICMIB 2011.

4. ©Springer Verlag.

   Chávez, E. and Finnie, G. TTL: A Transference, Transformation and Loading Approach for Active Monitoring.13th International Conference on Data Warehousing and Knowledge Discovery, DaWaK 2011.

5. ©HICSS 2013. (Conference paper accepted not yet published)

   Chávez, E. and Finnie, G. Using intelligent data sources to monitor unusual behaviors in individual's health data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and problem statement

In large organizations, massive volumes of data are generated in many heterogeneous and distributed data sources. Data changes rapidly and a large amount of information needs to be integrated to process data in a timely manner to provide the right information to decision makers. Therefore, a control mechanism is needed in order to support timely decision making in a constantly changing environment. To support this process, enterprise management systems specialize in the management of business information within organizations by combining Business Intelligence (BI) and Enterprise Content Management (ECM) to capture knowledge from an enterprise perspective [86][28].

BI applications put together processes related to data integration, data storage, and knowledge management with analytical tools with the aim to enhance the quality of inputs that ideally should arrive at the right time, right place, and in the right form to advise decision makers [111]. Typically, information in such applications is consolidated and analysed using data-warehouse (DW) technologies and federated systems. Most BI architectures monitor crucial business operations by using a data warehouse. In this way raw data from several and complex sources can be consolidated to provide high quality and consistent information [39][83][9][179].

In data warehouse environments operational processes are executed to move data from

operational sources to the warehouse. This includes data export, data preparation, and data loading processes that are usually performed by using Extraction, Transformation and Loading (ETL) tools. ETL functions are a critical component of a BI solution as they are responsible for retrieving data from different sources (one or more), preparing it by normalizing and transforming it in some way to then be inserted into some other repository usually called a data warehouse to be ready for analysis and reporting [152][56].

The main characteristics of ETL tools can be summarized as follows [151]:

- data identification at the source level
- data extraction from the sources (one or more)
- data integration into a common format
- data cleaning on the basis of database and business rules
- data propagation into the data warehouse/data marts

Traditional data warehouse systems may impose unacceptable delays as they are out-of-sync mainly because of their batch nature. ETL techniques have not been scaled up to address the challenge of data loading, performance and low latency to provide real time decision support systems. As an example, research reported by Adzic, Fiore, and Spelta [1] shows a data warehouse that has to perform the ETL process with a loading window of 4 hours in a system in which 80 million records per hour are produced on a daily basis. "The volume of data rises to about 2TB with the main fact table containing 3 billion records " [151, p. 7]. There is an evident need to improve performance to avoid extra steps when capturing, transforming and/or storing data.

The use of intelligent techniques for data preparation and loading may considerably improve the overall process of data warehouse population. Therefore, the design of back end intelligent ETL processes is becoming important but at the same time complex, as problems of optimization and modelling emerge. During the last decade a small number of research efforts were carried out to monitor data in real time and to address the challenge of ETL process optimization. Research reported in [86][181] [114] show a variety of approaches which

tried to break the gap between the time in which operational data is created and the time information becomes available for analysis. This is done by optimizing the ETL process, in other words by optimizing the execution time of the ETL process.

Research conducted by Javed,and Nawaz [78] presented a technique to distribute the Online Transaction Processing (OLTP) source data that has to be extracted to achieve real time data warehousing. The ETL refreshment is done off-line for some tables and for others in a near real time manner. They have then created a near real time data warehouse methodology. Moreover, in architectures like SARESA [114] which are based on an Object Oriented Architecture the authors argue that they have built a real-time warehouse architecture. The data changes are captured by means of data capture web services and real time data is pursued by using a multi level data cache.

After reviewing research conducted in real time and active data warehousing technologies for BI (explained more in-depth in Chapter 2), it is possible to say that; only near real time has been achieved, and most of the techniques studied focus on the "time" response from the "machine" point of view to deliver the information rather than on improving the speed to analyse data and therefore, provide advice within the right time to decision makers.

Data analysis is performed only once data has been consolidated, therefore reporting and alerts are the last activities after some sort of ETL has been done. Moreover, real time data warehouse approaches usually query and capture a vast amount of data, so data is pulled (extracted) from the local sources to then be processed in a consolidated repository. Furthermore, most applications are still human administered with no possibility of self-adaptation. The idea of dynamic adaptation to business requirements, i.e logical adaptation rather than structural adaptation, of a DW architecture has not yet been proposed. Consequently, the demand for BI solutions and new frameworks is continuously growing but information systems research in the field is limited and therefore, to put it charitably, sparse [111].

Most of the techniques already in use have to schedule in some way the data extraction and all of them analyse the data only when it has been consolidated in the data warehouse. DW has done a good job to pull together large volumes of historical data but they are usually not built to deliver real time information [44]. Although business intelligence architectures

refer to technologies, tools and practices to integrate, process, and present large volumes of information, there are no ETL (extraction transformation and loading) tools that can provide real time data processing efficiently. Therefore, there has not been proposed a mechanism that can continuously feed and update data warehouses as soon as data changes in the data sources [28] [114] [39]. Thus, traditional business intelligence and data warehousing technologies "do not directly address time sensitive monitoring and analytical requirements" [114, p. 77].

Typically, a DW is designed and built from historical information with the data being refreshed daily at best. Nowadays, the process to access up to date information requires that all the services and objects are connected to make data from the different data sources available with a minimum latency possible. Being able to respond quickly to business events denotes the difference between success and failure in areas such as the stock market, food manufacture, health care, logistics and others.

From the business side there is a need to capture data events as soon as possible to maximize the value of information [64]. Consequently a quick response from the monitoring systems can help decision makers to take sound decisions that can make a difference to respond within the right time to sell shares, to save lives because a risk scenario is present, to detect frauds, move products to quick sale and others.

From the research side, there is not yet a smart framework with some sort of adaptation to environment requirements capable of dealing with real time data integration from a variety of data sources and real time alerting. For this reason, the design of a real time adaptive framework that covers the process of predicting and responding to business requirements in real time to decrease the time of data analysis and enhance the value of information while using data warehouses is the focus of this research.

1.2 Aim and Research question

The overall aim of this thesis is to describe "*a smart and rapid response mechanism to analyse, process and monitor data to detect data abnormalities and/or risk scenarios in real time using data warehouses*".

The research question that this research study addresses is:

Does the local empowerment of data sources to pre-analyse and push data to feed data warehouses enable better response and alerting mechanisms to detect risk scenarios in real time?

The mechanism should be better in terms of:

Efficiency:

- Response time:
  - By reducing the numbers of steps to analyse data as a way to enhance the value of information and reduce analysis latency.
  - By identifying, monitoring and analysing meaningful data. Only the data that is necessary to analyse in real time will be monitored.
  - By using the knowledge of the sources to learn from the data (mainly historical) to know where and how to react and who needs to be alerted.
- Error detection:
  - By checking data content (type of data and structure) to detect abnormal behaviours, which will allow detection of data errors and inconsistency as soon as data arrives at any local source.

Effectiveness:

- to perform data fusion: by using a mechanism to pre-analyse data that comes from different sources at the local level.
- to detect abnormal behaviours while monitoring business activities: by having the ability to process local knowledge in a distributed architecture. Following the idea of smart sources a pre-analysis of meaningful data could be done at a local level to perform alerts and generate from the very beginning a chain reaction in the decision making process when risk scenarios are present.

Data content is what is analysed at the source level and it is represented by " *what the data looks like*". Content of a blood pressure reading must be composed, as an example, by

a numeric field such as 100-80,100/80 or 100,80. This reading "100,80" is defined as the data content that needs to be monitored at the source level. A unusual data behaviour occurs once the content of any data that has changed at a source is not within accepted ranges. Following the same example if blood pressure is part of the data to monitor, a reading of 200,100 might show an unusual behaviour given the usual reading should be under the limits of 120,100.

A risk scenario can be described as any unusual behaviour in the data that has changed at the source level that might represent a risk, advantage or danger depending on the situation, for the entity monitored. A risk in patient data monitoring could be a blood pressure reading to high or outside normal limits, whereas a risk for product monitoring could be an unusual number of sales which might be a positive finding that a product manager would like to know.

The effectiveness of the framework proposed will be assessed by testing it in the area of health informatics. A prototype solution to manage chronic disease patients is proposed by designing an architecture that learns and reveals the disease activity patterns through day to day measurements and the clinical history of an individual patient, reacts in real time by sending alarms according to changes in those patterns, and is adaptive to new system conditions and changes in health care requirements.

It is possible to divide the proposed research into two sections which are:

1. To deploy a real time data analysis and processing mechanism in order to accomplish the following objectives:
    - To propose ways to identify meaningful data to monitor,
    - To empower the data sources to analyse, process and/or transfer data in real time,
    - To set the rules and relationships of events and conditions that will trigger the data analysis and processing at the source levels or the data transference to the consolidated repository.
2. To enable an active monitoring mechanism capable of:
    - Identifying individuals data rules, main features and individual accepted data ranges. In this case identify individuals disease data characteristics such as normal and risk conditions to monitor for each patient with heart disease.

- Applying a monitoring mechanism to continuously look at patient data and learn when a patient's disease state has changed.
- Implementing a rapid response and alert mechanism when unusual patient scenarios are present.

As it was mentioned previously, nowadays real time data warehouse research approaches focus mainly on measuring real time from the machine point of view. The system proposed here focuses on achieving real time by analysing data and alerting decision makers as quickly as possible instead. Consequently, this research project has developed an architectural framework which demonstrates that it is possible to design a business intelligence model that by using local knowledge performs pre-analysis of data in distributed environments and therefore can monitor data in real time and pro-actively react to risk or unusual business scenarios.

1.3 Significance

Traditional data warehouse approaches have static architectures that are not capable of supporting changes in their content and structure. Data is refreshed periodically but they are not prepared for continuous data integration [143]. This is why real time data warehouses seem to be a feasible solution to address near zero latency. This can be done by decreasing the time among the processes of extraction, transformation, and loading of data into a consolidated repository.

In a competitive marketplace, organizations have the need to aggregate huge volumes of information and data inflows into useful parameters in order to take sound decisions in the right time which will subsequently enable them to maximize profits.

The major challenge is to provide an infrastructure that allows dynamic interaction between users and service providers in a timely manner. Real Time Decision Support Systems (RTDSS) monitor data and can shorten the decision-making process from days to minutes which might make the difference between success and failure for the organizations within a determined market [86]. RTDSS are usually built following a distributed system approach in which the main goal is solving a large computational problem such as data analysis and pro-

cessing from a variety of heterogeneous data sources. Therefore, it allows connecting several computers/applications enabling autonomy, local views and decentralization.

Nevertheless, the state of the art in distributed systems administration (data bases for example) is still dominated by human administration and intervention. A critical prerequisite for distributed database technologies to comply with the new challenge is that it be largely or completely self-tuning, provides data refreshing or preprocessing as quickly as possible and has autonomous adaptation capabilities to evolving environments with minimal human administration.

Monitoring systems have coped with the need for quick response mainly from the machine point of view. Approaches to shorten the time of data analysis/reporting and the use of intelligent techniques to perform pre-analysis and system adaptation so far have not been extensively explored. The main aim is to decrease the time it takes to make decisions to enhance the value of information in the organizations, but certainly no data can be really obtained in real-time. However, a system that keeps zero latency between the decision time defined for the service level agreement given by the organization deadlines, and closes the gap between cause and effect of these decisions, is needed [86][143][46]. BI systems do not seem to manage the integration of different and heterogeneous data well. [118].

Real time enterprise must be seen from the view of the implementation and use of a smart, active, intelligent, adaptive and minimum latency mechanism. The business value of time must be measured by the value lost while capturing data events, which is called data latency. The time spent while analysing these events is called analysis latency, and the time taken from reporting data findings until a decision is taken is decision latency. The more time spent to capture, analyse and deliver information the less value it will have for organizations [64].

Although the IT problem of providing real time data analysis to monitor data that comes from different data sources will be tackled, the research has been implemented in the area of pervasive healthcare monitoring by providing a patient specific system that, with the ability of local knowledge and adaptation in distributed environments, can actively monitor cardiovascular disease patients.

It has been estimated that without action, 388 million people will die from chronic diseases in the next 10 years. About 35% of them will be patients with heart disease. The macroeconomic impact could be substantial. In several countries, the application of existing knowledge has led to major improvements in life expectancy and quality of life. For example, death rates from heart disease have fallen by up to 70% in some countries such as Australia, Canada and the United Kingdom. Such gains have been realized largely as a result of the introduction of comprehensive and integrated approaches that encompass interventions directed at both the whole population and individuals, and that focus on the common underlying risk factors, cutting across specific diseases [122].

Cardiac disease monitoring of patients takes place at each step of patient management, from disease detection to disease prognosis and from surgery to recovery. During routine screening, day to day measurements (sensor devices) and patient knowledge abnormalities can be detected. Just in Australia Cardiovascular disease (CVD) is one of the leading causes of disability and death (17% of the overall disease burden in Australia in 2003). "CVD is the most expensive disease group, in terms of direct health care expenditure,at over $5.9 billion, which represents 11% of Australia's total allocated health system expenditure in 2004-2005" [108, p. 10].

In a scenario in which real time data monitoring is needed several issues arise such as complex distributed data processing, data fusion, communication and uncertainty of data analysis [93]. Real time decision support technologies enable doctors, patients and the health care staff to respond to a crisis and prevent problems by providing early indications of deterioration in the patient. Thus, one solution is to design an architecture that learns and reveals the disease activity patterns through day to day measurements and the clinical history of the patient, reacts in real time by sending alarms according to changes in those patterns, and is adaptive to new system conditions and changes in health care requirements.

1.4 Dissertation outline

The subsequent sections of this dissertation are organized as follows: Chapter 2 provides the literature review by examining the current state of the art of real time decision support sys-

tems, business and health care data monitoring. Chapter 3 discusses the methodology used to conduct this research. Chapter 4 describes the framework proposed and its design. Chapter 5 presents the prototype implementation undertaken to validate the concept proposed. Chapter 6 shows the framework validation in terms of efficiency and effectiveness. Finally, Chapter 7 summarises the main findings, conclusions and future research.

Chapter 2

Literature Review

2.1 Introduction

This research involves the elicitation and analysis of the key factors associated with a real time decision support framework, that uses a data warehouse architecture and enables real time data management through the use of intelligent data sources.

An overview of the areas involved in this research can be seen in Fig.2.1.

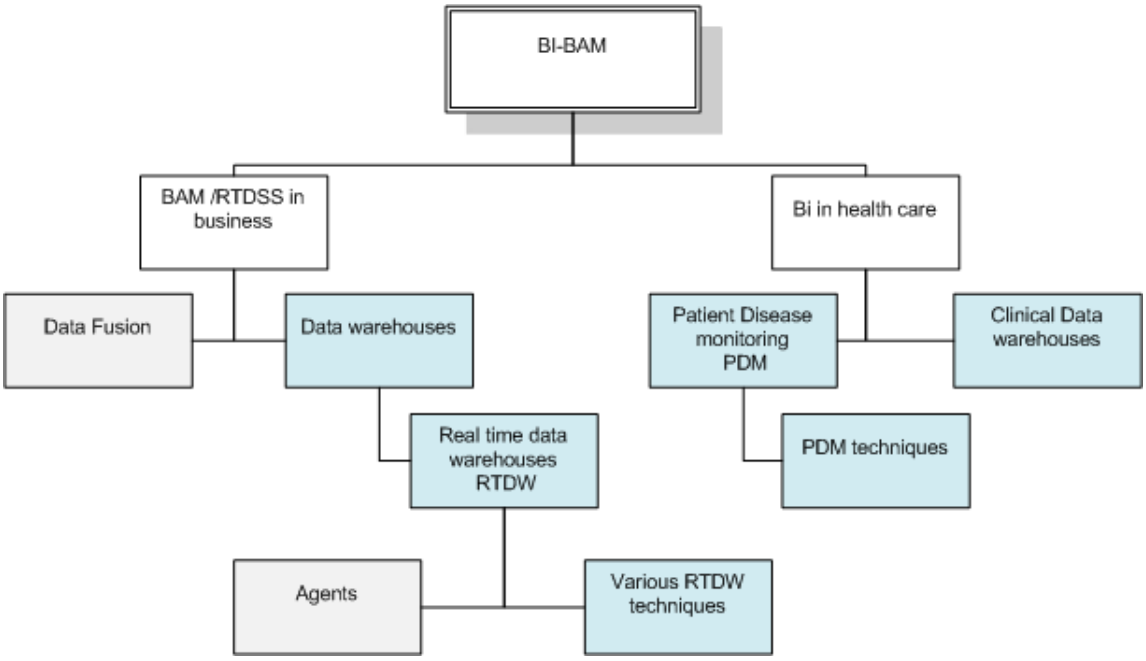

Figure 2.1: Areas involved in this research

The framework proposed has its basis in the business intelligence domain. The main focus of the framework is to extract as much information as possible from a variety of data sources to convert it into knowledge to monitor relevant organization data in real time. Business activity monitoring (BAM), a part of BI, by using DW aims to provide an architecture for monitoring time-critical operational processes to support sound decision making.

In Fig.2.1 the grey boxes represent the areas that the current research touches but will not be discussed in depth as they are not a relevant part of the research questions. In the blue boxes are the dominant themes of the research question such as real time decision support systems, real time data warehouses and the techniques already in use to support the implementation of real time data analysis by using data warehouse architectures. As a way to prove the research concept proposed a prototype in the health domain was built. Therefore, a view of the current research in BI and e-health was also needed. The research gaps and current research in all these areas are discussed and presented in this chapter.

In today's information era organizations must be able to integrate different and large amounts of data that come from a variety of heterogeneous sources such as operational systems, sensors, other people and organizations, in order to support tactical IT plans and strategic decisions. Business Intelligence seems to be the right paradigm to follow to help managers to make timely and effective decisions. BI tools help with information gathering and processing data, building rich and relevant information that is then fed back to decision makers [39].

Most BI architectures use data warehouse technologies as the way to consolidate, analyse and report data that comes from different sources. However, although research in the past has treated data warehouses *as collections of materialized views* whose data is regularly refreshed and locally stored [151], today requirements have changed and real time transactions are required to support on-line operational decision making. Therefore, new approaches have been proposed to deal with data analysis as fast as possible to perform timely decision making.

An overview of the current research that aims to overcome these new challenges is presented in the following sections.

2.2 Real time decision support systems and active monitoring

Processes and business activities need to be maintained and improved to ensure that the organization is under control. One of the solutions that allows organizations to detect possible risks, to identify problems associated with critical operations in an executing business process, and also to recognize opportunities at an early stage is Business Activity Monitoring. BAM is an extension of business intelligence solutions, and involves near real time reporting, analysis and alerting of relevant business events [158].

BAM aims to deliver real time information in rapidly changing business environments by providing information about the conditions and outcomes of various business operations, processes, and transactions. BAM focuses mainly in delivering real time performance information by using key performance indicators. Therefore, high levels of key performance indicators govern the process of monitoring in order to achieve a real-time view of a company progress. Data from the processes and business performance, such as metrics, must be continuously collected. As the monitoring processes of data are closely related with business performance a large amount of data is frequently generated [83].

BAM main components usually include [56]:

- A right time integrator which integrates data from the sources to the DW in right time.
- A repository capable of storing short term data for fast capture.
- A performance indicator manager.
- Some set of mining tools.
- A rule engine that continuously monitors and generates alerts.

One of the challenges of BAM architectures is dealing with data latency, which is the period of time between the moment an event occurs and when it is known by the decision maker [56]. This implies the adoption of new techniques which raises problems such as data integration and quality [83]. BAM research and solutions to react in near real time can be found in [72][105] [114][126] and [158]. In these, data mining models are trained to specifically analyse data and to look for particular conditions by triggering internal and

sometimes external actions that would indicate trouble and/or opportunities. Actions include alert with recommendations, alert with guided analytic, and automatically send messages to one or more internal applications to cause something to happen as a whole chain reaction. Nevertheless, in BAM as a system reacts or alerts most of the time only after detecting specified conditions it might be too late, particularly when the conditions and rules come into sight at the end of a process [84].

As a result, depending on the situation it is difficult to "pro-actively" check process performance to take sound decisions in real-time. Thus, giving on time process states requires not only gathering the information in real time but also performing some intelligent analysis to recommend an action to know what to do next, as well as reporting that information in a timely manner as the longer the time to alert the less value that the alert will have [9][181]. Golfarelli et al. [56] envisages that in BAM the use of alerting interfaces with a static structure to achieve right time freshness could work better with a push approach to react to business events.

Dayal et al. [39] give an overview of the main characteristics to consider in the next generation of BI architectures (See Fig. 2.2).

In current BI solutions the data flow is mainly one-way, from operational systems to the warehouse. In future architectures BI should support bidirectional flows, thus after the data consolidation the data warehouse would be able to provide feedback and updates to the sources of information, and customer data interfaces computed on the fly would also be possible to implement. Additionally, at a physical level, the next generation of BI solutions that uses data warehouses should consider the adoption of alternatives other than ETL such as hybrid styles or Extraction, Loading and Transformation (ELT) [39].

Even thought not all the problems to achieve active monitoring in real time can be solved at this stage, it can be seen that there is a need in large organizations to make up-to-date data available on a real time basis to minimize the time to perform data analysis, to look at the data that is changing, and therefore perform alerts as quickly as possible when risk scenarios or complex situations that affect the organization are present [179][40].

To support real time active monitoring, real time decision support systems (RT DSS)

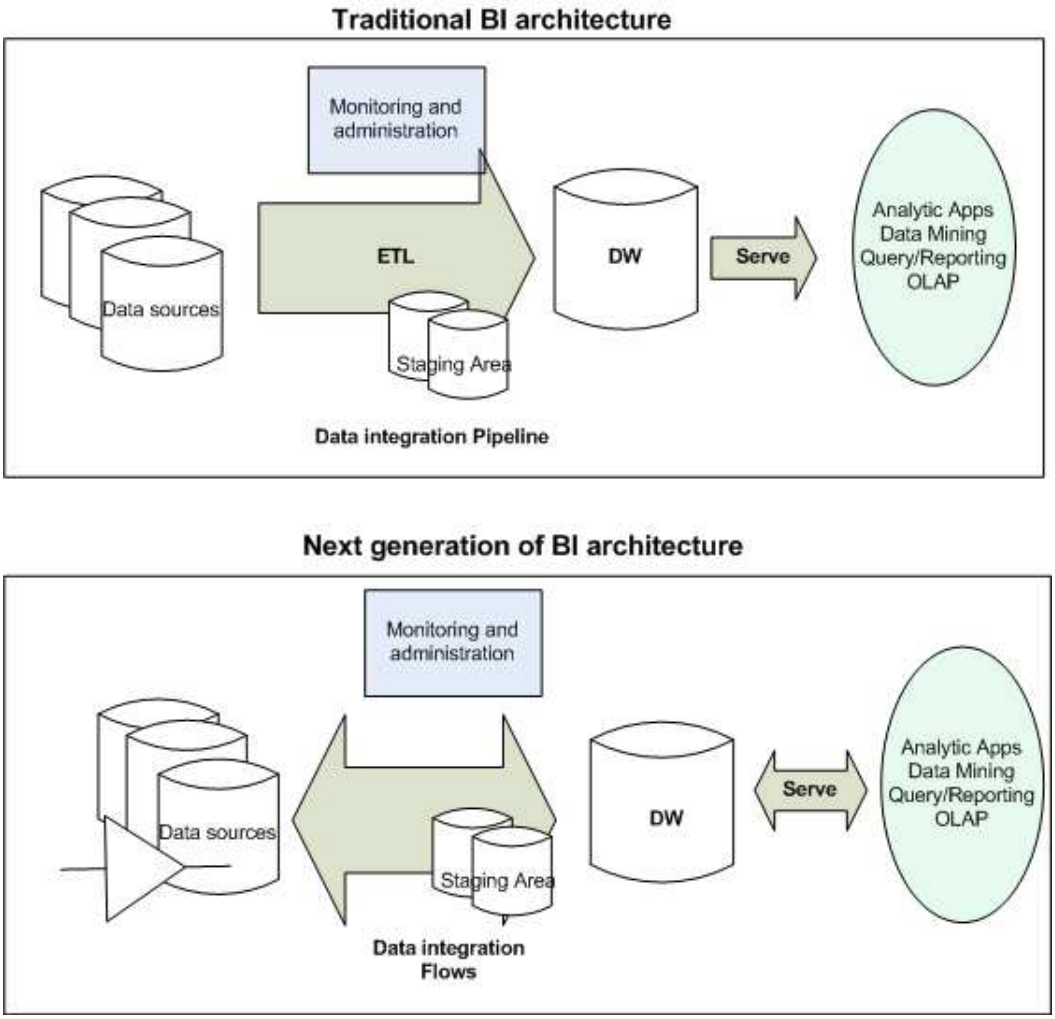

Figure 2.2: Next BI vs Current BI architectures by [39]

incorporate different technologies such as enterprise data integration applications, real time systems and data warehousing. Some of the problems to be addressed in the implementation of such architectures are the integration of heterogeneous and distributed systems and the possibility to analyse and process information in the right time.

Nowadays it is not as difficult, as it used to be, to capture all types of data and store them cheaply. Data can then be accessed and stored quickly. However, "real-time data analysis and actions still seem a remote reality" [9, p. 1].

As in BAM architectures, RT DSS aims to support timely insights into business parameters and trends by gathering accurate data from different operational systems into useful information. Even though response time is important, users need business operations re-

ports to be up to date too. RT DSS applications usually require three main categories of technologies: data warehouses, analytical tools, and reporting tools.

RT DSS tend to predict outcomes based on the information gathered (knowledge). This prediction is then explained in a set of recommendations or actions to be taken in consideration by decision makers. It can be seen that the main areas of research in RT DSS focus on dealing with problems of data integration, application integration and on delivering the transformation of important data into valuable information [179][86][166].

Nowadays, RT-DSS are part of integrated intelligent/expert systems that use knowledge bases to support real time decision making. It can be seen that the trend to active monitoring is based on the design of agent layers to help monitor and manage business operations at different levels in the enterprise (Fig.2.3). Thus, through collaboration and interaction among different agents, autonomy and flexible architectures have been proposed in order to handle unexpected exceptions in different monitoring scenarios (A review of agent technologies used in intelligent decision systems is given in section 2.5).

Given the use of intelligent agents for active monitoring a special section to discuss it will be presented in section 2.5.

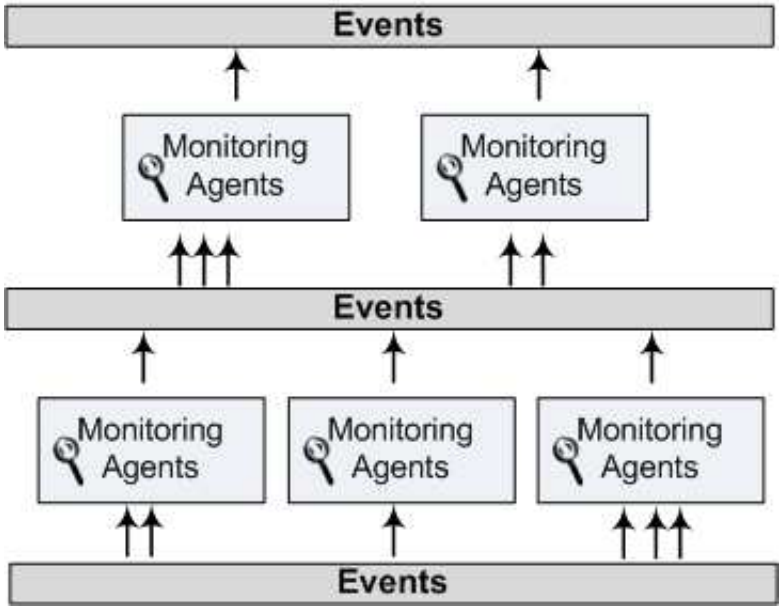

Figure 2.3: BAM Trends - Layers. Adaptation of [51].

RT DSS architectures layers are mainly:

- The semantic adaptation layer which is most of the time implemented with a message broker mechanism,
- The integration layer which is mainly implemented using federated DB or data warehouse technologies
- The learning and analysis layer which includes the data mining mechanism to identify patterns and offer possible actions,
- and the reporting layer which provides the necessary views for the decision makers [86].

To achieve a certain level of adaptation in RT DSS, research in [105][9] and [51] presents successful solutions with the use of encapsulated software-based systems. These systems use agent oriented techniques which are appropriate for complex and distributed software systems. Agents were designed to perform specific goals and to learn about process capabilities to automatically build up learning models and are used to answer what if questions and to respond to particular events. Intelligent agents are mainly proposed in RT DSS to manage business activities capturing events and monitoring the state of work flow tasks, processes and resources.

Moving into the area in which researchers have focused in measuring real time from the organizational point of view, a novel approach to measure the value of information in BI was given by Yan et.al in [179]. Yan et.al research maximises the information value of query processing rather than machine response time stand alone. By using a value-driven query processing the researchers measure response time as well as the time that lapses since a query is made till the information is placed in a report. Consequently, each report generated has a time stamp and is assigned a business value. Information value is then the value of a report discounted by time, and the risks associated to computational and synchronization latency. Thus, this research highlights the relevance of response time but also recognize the business value of the information from an organizational point of view.

According to Hackathorn [64] the value of information from a business perspective depends of data latency, analysis latency and delivering latency, which means the more latency since information is gathered and analysed the less value the information gathered will have for an

organization. Thus, real time might must consider not only fast data integration, but also the identification of what is really needed to be captured promptly and the decreasing in time of data analysis to provide sound reports/alerts to decision makers.

In most of the real time approaches reviewed and previously cited, the data updates and the analysis of the data is usually performed according to system administrator criteria which defines the events and conditions in which a query or data request is necessary, and checks the incompleteness, uncertainty and contradiction of the data designated for analysis. All the systems studied monitor the data content only after it has been integrated in a consolidated repository.

Current approaches focus on capturing data quickly by querying the sources, using query optimization, and /or using streaming data techniques to move a vast amount of data rapidly from the sources to a central repository. *Data analysis and reporting is then left as a final step after the consolidation process has been done.*

Thus, a re-distribution of the ETL processes might decrease data analysis latency to provide decision makers with the right alerts/reports in a timely manner. This research aims to show that and consequently it includes the analysis and design of a framework that enables data updates based on the knowledge that the source systems already have and from day to day activities and business processes. Based on the source's previous knowledge of the data stored, a pre-analysis of the relevant data to be consolidated in a warehouse can be achieved. Thus, by decreasing data analysis latency, detecting at an early stage any unusual behaviour in the data collected, alerts/reports are provided to decision makers in a timely manner.

To provide a big picture of the main problems in the areas previously mentioned to achieve active data monitoring using a data warehouses a more extensive discussion is presented in the following sections.

2.3 Data warehouses in BI

Business Intelligence (BI) aims to collect, integrate, analyse and present large volumes of information to enable reliable and timely decision making. To achieve that a BI architecture commonly includes the use and implementation of a data warehouse [39].

As a concept a data warehouse (DW) should be able to store a large amount of data integrated from different and independent data sources [27]. In general a DW architecture is built on top of existing operational systems and should comply with the following four characteristics [75]:

- Subject-oriented: All significant information is gathered to analyse data according to particular areas of interest usually called the subject areas or domain. It represents what the company wants to have information about, shipping finance and others.
- Integrated: Data is stored in a global and common manner even when the underlying operational systems store the data differently.
- Non-volatile: The information is stable and consistent, there are no changes, inserts, or deletes performed against the historical data once it has been entered into the warehouse.
- Time variant: A DW needs to change over time as it maintains both historical and (nearly) current data.

DW is a meaningful component of data-driven decision support systems (DSS). They provide decision makers with a business-oriented view of data by relying on multidimensional models. DW tools help to report a consolidated reality of current business operations so then decision makers might discover new trends and business areas of development [132][27].

There is no standard in data warehousing design methodologies. However, data warehouse implementation involves activities such as data sourcing, data staging (extracting, transforming and data loading), and the development of decision support systems oriented to end-user applications [145][9][91]. The value of these activities will depend on factors such as data quality management and meta data management.

In a traditional data warehouse architecture data is integrated into the warehouse in three steps; Data is extracted first from the sources, then is transformed and cleaned to ultimately be loaded into the warehouse. Extraction, Transformation, and Loading processes (ETL) are the key to move and consolidate data from a variety of heterogeneous data sources, to a homogeneous environment, in a single repository [78]. These processes are time consuming and in most cases crucial for DSS implementation success.

ETL tools functionalities include [151]:

- extracting data from a variety of data sources and/or applications,
- customizing and integrating all this data to a common format,
- transforming the resulting data set according to business and data base rules,
- and finally loading this data to the data warehouse.

Traditional data warehouses are refreshed in a periodic manner (batch-based updates), usually on a daily basis (off-peak hours), where the operational sources and the data warehouse experience low load conditions. Therefore, there is a retardation between business transactions and their representation in the DW, the most recent data is unavailable for analysis as it is caught in the operational sources [77]. Although for some companies this could be suitable, there are others that require fresh information available on a real time basis and have the need to perform the analysis with up-to-date data. Hence, it is possible to state that traditional data warehouse technologies can be out-of-sync very quickly which can be an issue in obtaining real time information response [179].

As Chountas and Kodogiannis [31, p. 1] state, "there is an urgent need to be able to aggregate this massive amount of information in a useful way". Active Data Warehousing (ADW) and real time data warehouse applications are required in which large amounts of heterogeneous information can be updated as frequently as possible [160].

There are successful examples of dynamic multidimensional models [8][31] [127] and [141], whose research has used dimensionality reduction techniques, data mining or On-Line Analytical Processing (OLAP) capabilities to enhance the warehouse functionality. Nevertheless, these applications, as in business warehouses, have been developed to make decisions using a large amount of historical information and executing the information refresh in large batches (e.g. once per day). Thus, applications which enable real-time analytics across business processes are issues pertinent to future study.

It is said that around "80% of a BI project cost is being attributable to the design and development of the data warehouse" [37, p. 3]. Therefore, it is really important to choose the right architecture and tools to design and build the DW. Until now ETL seems to be

the "standard". Nevertheless, as [170] discussed in some cases, technical reasons, such as the need to handle extensive processing to perform complex transformations of data, and business reasons, such as the need to include new requirements in the architecture, have caused new approaches to be considered.

As ETL activities are relevant in the design and development of a RT DSS using data warehouses an overview of current research in this area is presented in the following section.

2.3.1 Extraction, Transformation, and Loading processes

Extraction, Transformation and Loading processes (ETL) are the bridge to move data from the sources to the DW. At present, the dominant method to move data is based on pulling data periodically. Consequently, ETL usually works over a batch of data sets of business data, pulled from the sources at a given time such as a day, week and some times a month [21][183]. Information is then available for analysis only after this batch refreshment has been performed. For some organizations this might be acceptable but for others just in time analysis is needed.

ETL phases can be explained as follows [3][150]:

- The Extraction phase consists of reading and identifying the data from various sources, mostly transactional data. It comprises the technical problems of dealing with capturing and importing relevant data from heterogeneous sources into a staging area.

- The Transformation phase, the central part of the ETL process, brings data together into a common data model (format) by using techniques such as data mapping/duplication technologies and data cleansing techniques. Here data syntactic and semantic problems arise. Usually, data cleaning is performed in a different area called data staging.

- The last stage, the Loading phase takes all the cleaned, integrated and consolidated data from the staging area into the warehouse.

Data is propagated from the sources to the warehouse in a pipeline fashion, the central repository then pulls data from the OLTPs. Thus, the DW maintains historical records of

data as a result of the transformation, integration and aggregation of the data taken from the sources [170].

An overview of traditional ETL processes using a staging area can be seen in Fig.2.4.

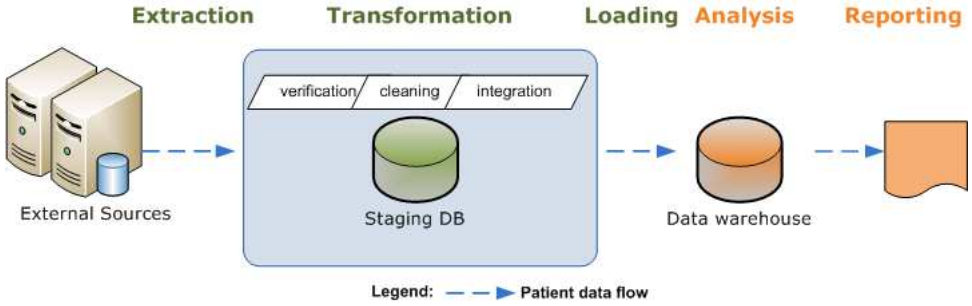

Figure 2.4: ETL mechanism

There are several ETL tools such as [73][4] that allow data consolidation from commonly one or more heterogeneous data sources into "a single physical store". This approach has proven to be one of the most effective solutions to provide quick access to relevant but widely dispersed information [3]. Nevertheless, as has been mentioned previously, there are new requirements such as the need of business users to have a higher level of data freshness, new type of sources entering into the scene (sensors and the web), better monitoring and the demand for accurate reports with up to date data [170] which have made the need to consider the possibility of achieving near or real time data freshness in the data warehouse.

In terms of loading techniques a DW can be populated in two different ways, a traditional way called *Bulk Load* or for some in a more efficient way called *Incremental Load*.

1. Bulk load, some times also called full load, refers to a DW refreshment typically performed in batch mode on scheduled basis. Information is extracted from the sources, the resulting data is collected and then compared with the data warehouse content. In this way, the required changes for data warehouse refreshment can be retrieved. "Full reloading is obviously inefficient considering that most often only a small fraction of source data is changed during loading cycles" [81, p. 2].

2. Incremental loading focuses instead in capturing source data changes to then be propagated in an incremental way to the warehouse. This type of technique is the one most

commonly used to achieve real time in data warehouse environments.

Traditional ETL tools do not seem appropriate to be used to actively monitor data in real time [17] [143] [21]. It is necessary to minimize the latency of the ETL process and therefore, a review of the current research in the area of active data warehouses is needed. So far, most of the near real time data warehouse approaches focus on optimizing/shortening the loading cycles or speeding the ETL process. The major challenge here is then to work with the implementation of near real time data refreshment to load data as quickly as possible. A more in depth discussion about incremental loading techniques such as Change Data Capture (CDC), for real data warehousing will be discussed in the following sections.

2.3.2 Near real time data warehouses

Certainly, no data can really be obtained in real time, not in a *quantum sense* because by the moment data is seen it is no longer real time [133]. For some researchers real time means *up to time* or *right time*, and it could be describes as any data change that is taking place in a source system that has an immediate and automatic echo in the data warehouse [164]. For others real-time is not about being fast, it is the utility function that designates the damage for the organization from missing the deadline [46]. This means the function represented by the value of information ETL latency, which could be represented by the time a business event is captured until the time that information is ready for analysis [44]. Therefore, it involves some qualitative and quantitative rules as Nelson,and Wright [113, p. 111] states " The critical challenges of the decision support in general is how quickly can we make sound decisions? The issue really revolves around time to decision".

This means real time for an organization could be considered as the ability to respond to a decision in a day due to it crossing the overnight-update barrier, or the ability to make data flow without delay (trickle-feed) instead of batch load; or in a practical sense, real time will be defined by the *service level agreement given by the organization deadlines* (e.g. ability to report and fix a problems - within 2 hours for electricity companies or the speed to deliver food- no more than 20 min within 20km)[46].

In other words, real time must be related to how much time a company is willing to wait

to solve or to react to a particular event. Therefore, a minimum latency should be tolerated (See Fig.2.5) and the real time value will depend on individual business organizations. Some organizations and in particular some processes to be monitored, will be willing to tolerate more data analysis latency than others. But in general for critical operations *the more time to capture an event the less value for the organization the data being captured will have* [64].

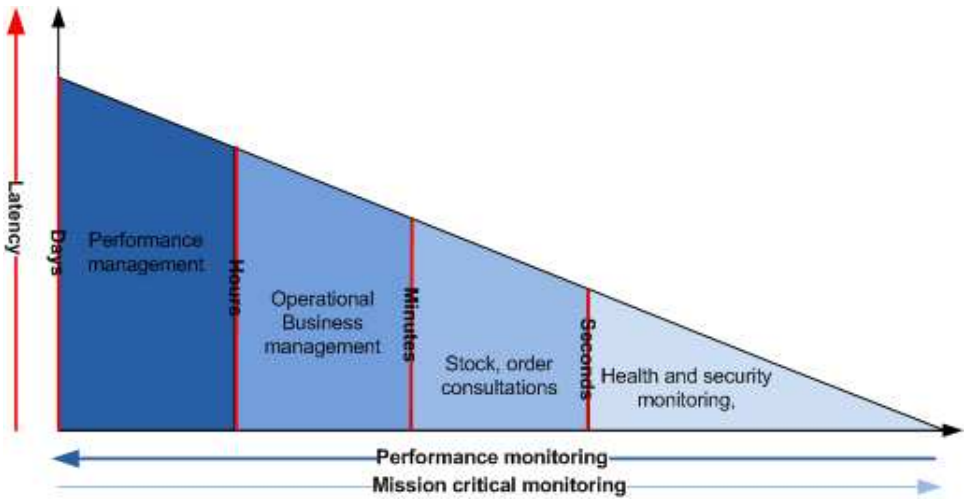

Figure 2.5: Monitoring needs vs data integration latency. Adaptation of [62].

Real time data warehouses refers to a trend in which the information consolidated for the DW is available for analysis as rapidly as possible. It aims to decrease the time to make business decisions by minimizing the cause and effect of business decisions [17]

Different approaches, methodologies and technologies have been proposed in each of the stages of the DW design to achieve real time [91]. Some focus on continuous data integration such as trickle feed while others focus on optimizing the ETL processes. The main approaches are:

- *Continuous data integration*
  - *Trickle Feed*, in which the data warehouse is continuously fed (loaded) with new data from the source system.
  - *Trickle and Flip*, in which the data is continuously fed into staging tables that are in the exact same format as the target tables. It helps with issues such as tables being simultaneously updated.

- *Real time partitions*, in which copies of fact tables are made to keep the last updates.
- *Real time data cache*, which can be a dedicated database server or another instance of a large database system with the purpose of loading, storing, and processing the real-time data.
- *ETL optimization* query optimization and speeding the loading process by using data streaming are the main techniques used to pursue real time.
- *Variations of ETL to achieve real time* A few approaches have been proposed to achieve real time by changing the states, order and actions to transfer data from the sources into the warehouse.

A better understanding of these real time approaches and research, showing their advantages, disadvantages and functionalities, is presented in the following sections.

2.3.2.1 Trickle feed

Trickle feed solutions are mainly used in finance where stock prices or currency exchange rates that change during the day are loaded as they change [163]. It attempts to continuously feed the data warehouse with new data from the sources [91]. This technique uses streaming to achieve real time data loading by setting intervals to commit after a specified number of rows have been loaded, after a determined time period has occurred or by using Change Data Capture (CDC) mechanisms. Most of the architectures implemented though work under a messaging infrastructure via streaming data.

Many systems and models have been built to tackle stream data management such as the one described in [24][9] [51] and [61]. Event stream processing or complex event processing are paradigms which are helpful to react in real time to changes of states by correlating meaningful events within the data being streamed. Nevertheless, commercial applications are still limited in achieving data integration and analysis in a timely manner and in handling changes in data rates and distributions. Research in this area mainly focuses on "on-the-fly computation of queries" [151, p. 1].

Change Data Capture, one of the recent approaches being proposed as part of trickle feed solutions for incremental loading, through the detection of data that has changed at the source level, has lately created greater interest. This approach seems to be very promising as it integrates and captures data based on the identification of changes in the operational sources. Change data capture is mostly deployed to be used in data warehouse environments but since it captures and preserves changes of data across time, it can be also also applied to any database system.

The latest vendor solutions for BI, such as [140] [120], include a change data capture mechanism. A typical CDC package uses a publish and subscribe mechanism where changes at the source level are processed and published into a group of tables. Change data capture solutions needs the use of a staging area, and usually must be developed under the same release package, hardware and operating system.

Research findings in [162] [154] and [77] argues that CDC makes data extraction more efficient by processing only the changes made at the source level. Changes in relational tables can be identified by using triggers, status, version numbers and timestamps in rows. Most CDC approaches used a publish and subscribe mechanism where changes are captured and placed/published into change tables. Consequently the subscriber can then query the change table.

It can be seen that using CDC the amount of information transferred into the data warehouse is reduced which maximizes speed, and by filtering the data first (sending only data that has changed) it could be said that resource consumption is being minimized too. However, disadvantages in using or implementing CDC were encountered because to really achieve real-time data integration, changes have to be captured as part of a transaction and for "some CDC techniques there is some latency between the original change in the base relation and the change being captured"[104][6]. It needs to be considered then, that a transaction in an operational database can have one or multiple events and therefore, all the events for a particular transaction must be captured.

Changes are generally published or transferred to a table, where a subscriber, usually a Data Base Administrator (DBA), works and analyses them to decide what to do next. In

other words changes, even if they are placed into tables or logs, have to be queried by the warehouse. These changes are not analysed until they have all been consolidated in the central repository so once again it is the data warehouse which drives the data capture and analysis. The same traditional approach is followed here but just with less data being transferred to the warehouse.

2.3.2.2 Trickle and flip

Applying the trickle and flip concept does not seem to be very complicated. Instead of continuously loading data directly into warehouse, data is loaded in a staging area. Staging tables are replicated and on a time basis exchanged with the real-time tables [184]. Consequently, data is not loaded to the actual tables, it is fed instead into staging tables which are the same format as data warehouse tables. As new data trickles in, as one or more records, they get immediately attached to the fact table copied. Then on a periodic intervals (say hours or minutes) the trickle is halted, the staging table is copied, renamed and swapped with the fact table which brings the data warehouse up to date (See Fig.2.6) [91][155].

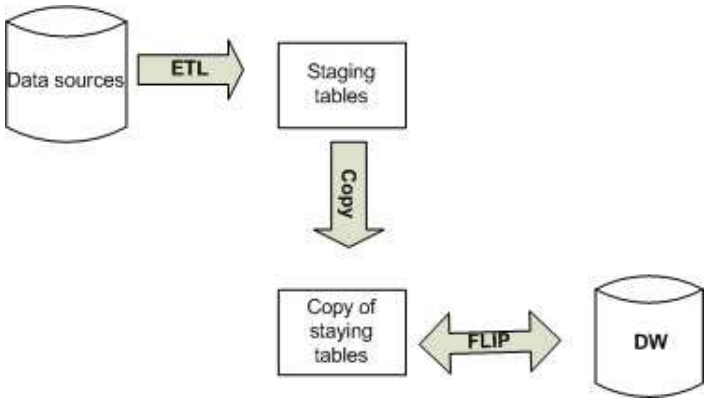

Figure 2.6: Trickle and Flip [184]

The problem of this approach is that it is not scalable [155]. Hence it causes problems when a huge amount of data needs to be moved and the fact tables are very large so the time to copy them can be too slow and therefore increases latency [134][184]. Consequently, to date there are not too many studies that apply this approach and partitioning methods have been used instead.

2.3.2.3 Table partitions

The idea of table partitioning is to use the features of relational databases to create table partitions. Thus, a criterion is specified, usually a time criterion, to decide when a new partition will be created [155].

Real time partitioning allows the creation of large tables that will be handled internally by the database as a series of smaller ones. In other words, new tables that resemble the active fact tables are created and designated for quick updates. These tables, called interval tables, contain data from only the last updates [134]. The real time data is then stored in the latest partition.

A disadvantage can be creating the right criteria for partitioning as a vast amount of data will create too many partitions. Therefore, the database management system has to support this functionality which again causes scalability problems. It was not possible to find research that handles this problem, as in trickle and flip, real time table partitions do not seem to be used too frequently as real time approaches to manage vast amounts of data to load/feed data warehouses [155].

2.3.2.4 Real time cache

A real time cache is a mirror of the sources that handle real time data. The main idea is having tables in the data cache database that resemble the original data warehouse tables. However, just those tables that hold the most recent updates and contain real time data are used to build the data cache [114][155].

A real time data cache approach for a real time business application called SARESA has been proposed by Nguyen, Schiefer, and Tjoa [114]. This research tried to close the gap between the time in which operational data is created and the time in which the analysed information becomes available. Systems are fed with data from the operational environment. Then for the continuous analysis requests, data is managed by sense and response loops which are the steps that cover the process to sense, interpret, predict, automate and respond to the example scenarios. To prove the SARESA concept a prototype was developed for mobile fraud detection in which business information was continuously generated and decisions were

made triggering system responses such as sending alarms [114].

The problem with most of the solutions that implement real time date cache is that it is not possible to join reports and co-display alerts to display real-time and historical information together. Therefore, this type of solution has shown more efficiency when historical information is not needed. Moreover, if complex analytical reports are run on the real-time cache, it is possible for it to start showing the same internal report inconsistencies, database contention, and scalability problems the two previous approaches have presented [91] [155]. Some approaches are working on solving scalability problems by working with multi level cache such as in [183][94] but so far no implementation to actually prove near real time data warehousing in practice was shown, given near real time was validated by using theoretical approaches only.

2.3.2.5 Variations of ETL

Extraction, Loading and Transformation (ELT) is one of the emergent approaches that aims to reduce the cost of ownership in terms of maintenance of the ETL process [37]. In ELT data is extracted from the data sources and then loaded into the warehouse. It seems that there is no accessible research in ELT designs, nevertheless vendor companies have published a few white-papers to discuss advantages and disadvantages of using this approach [37][101]. An overview of ELT can be seen in Fig.4.

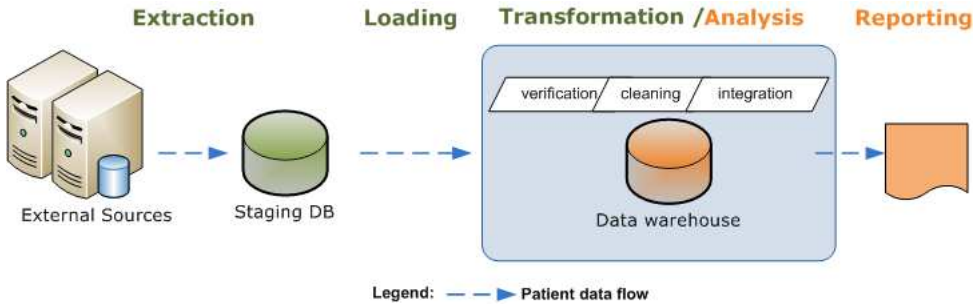

Figure 2.7: ELT mechanism

Benefits of ELT can be that it takes less time to get the data from the sources and transformations are only performed once data is in the warehouse. This cuts out the need to aggregate the data that has changed on a centralized server and removes an "intermediate

step from the overall data flow", as well as the associated costs of ETL server managing [37]. Disadvantages of this method include the fact that transformation will use extra resources for execution from the data warehouse and error detection from data cannot be discovered easily and on time therefore, database roll-back will be needed if an error takes place on a temporary table while performing the transformation process.

Both ETL and ELT are still humanly administered for tasks such as data updates and some parts of the analysis. Moreover, in both approaches data analysis is done only once all the source data has been loaded in the warehouse.

The research of Chieu and Zeng [28] offers *CTU* a capture, transform and update mechanism to incrementally update the performance of the warehouse in real time. This approach uses data triggers as the main components to initiate the sequence of actions to push data and stores daily real time data in separate warehouse tables with the same structure (partition). It works with an incremental data capture algorithm to detect data that has changed at the source level. However the researchers agree that inconsistency and scalability problems were found as a result of the implementation.

It can be seen that most of the techniques already in use have to schedule in some way the data extraction, and all of them analyse the data only when it has been consolidated in the data warehouse. There is no approach to date that pre-filters data and checks for data content rather than the time (logs or timestamps) to detect data that has changed at the source level.

2.3.2.6 ETL optimization and work-flows

During the last 5 years a few studies in the optimization of the ETL process have been conducted. Research in ETL work-flows argue that the use of pure streaming techniques are not enough to deal with real time using ETL.

Thus, most of ETL work-flow approaches focus on connecting business need/events to ETL rules [151][21]. Existing studies focus on logical optimization of the ETL process such as Simitsis, Vassiliadis, and Sellis [151] who proposed a framework to optimize ETL processes by modelling the problem as a state space search problem (in which activities are placed

in the flow). However, concurrency and the real application of the approach proposed was not discussed in the study. This study can be used mainly as an indicator to identify the challenges and issues while using or manufacturing a work-flow that implements the loading of data in a warehouse.

Even though a vast number of approaches have been developed to address real time data management, there is still problems related to real time data analysis and monitoring using data warehousing architectures not fully resolved. A few approaches have been implemented and discussed and most of them primarily focus on pushing data and speeding the loading process of ETL only. Data Fusion is the other area of research to consider as it is a important part of any data management architecture.

2.4 Data Fusion

Heterogeneous data source integration tasks involve querying the sources, combining the results and presenting them to the user. There are at least three steps to be performed in an integration scenario: Transform the data in the sources into a common representation, find data duplication, and combine all the data into a common representation. This last step is known as data fusion [15].

Data fusion (DF) aims at fusing a variety of records to represent the same real-world object into a single but clean and consistent representation [14]. The main idea is to have a common interface for an application that has access to a variety of sources that are physically distributed and heterogeneous (different data specification languages, data types and structure)[58].

Data inconsistencies/conflicts is one of the main areas of research in fusing data. There are different strategies to manage inconsistencies such as conflict ignorance, conflict avoidance and conflict resolution [42][14]. There are a number of strategies for fusing data into one consistent representation such as Pass it on, Roll the dice and Keep up to date [110]. Nevertheless, efficient query processing in distributed information systems remains an issue as the query processing framework needs to be designed to manipulate a small but potentially complex set of relevant queries required for the decision making process [181].

For years research in data fusion was mainly focused on relational data base management (DBMS) systems while nowadays MAS architectures seem to be more appropriate given the ability of agents to work as a one entity through cooperation in distributed environments. Thus, MAS architectures in data fusion have the capability to emulate traditional architectures, and provide useful services for system navigation and information integration [49][63].

Although data integration is relevant in the area of active monitoring in real time, in the current research it will not be discussed in depth as other aspects of data consolidation from a variety of data sources such as data analysis and loading were considered more important to study to solve the research questions.

2.5 Agents

Given most monitoring architectures use software agents, it is important to include a short review of agent technologies as a tool to active monitoring in BI. Agents are entities within an environment, that sense and react according to certain attributes considered in a particular domain. Agents might be capable of controlling actions across multiple technologies, extracting information and communicating with other agents to react to certain environment inputs.

There are different types of software agents depending on the environment and amount of information made explicit and used in the decision process. These can be intelligent agents, reflex agents, goal based agents and utility based agents [139].

Software agents main characteristics are flexibility and the ability to deal with partially observable environments [139]. Flexibility can be seen as the agent capacity to accept new tasks (new goals) and to update relevant knowledge by being told or learning new knowledge in relation to changes in the environment. A knowledge based agent can also combine general knowledge with current perceptions to infer a future state before taking an action. Thus, the agent can use its base of knowledge and some association patterns to infer a future state.

2.5.1 Software agent main characteristics

There are a large number of definitions given by researchers that state the attributes that an agent should have [13][54][139][157]. Those include attributes such as *adaptable* as the ability to learn and improve with experience (part of the flexibility attribute described in the previous paragraph), *autonomy* as the capability of self-starting behaviour, *collaborative* as the ability to work with other agents to achieve a common goal, and *reactive* as the ability to selectively sense and act about the environment.

The agent ability to cooperate makes possible the development of Multi-Agent Systems (MAS), in which agents work together to solve problems that exceed individual capabilities. A MAS is composed of multiple components, each of which has partial capabilities to perform a task. MAS have the characteristics that there is no global system control, the information is decentralized and usually computation is asynchronous [2][157][105].

MAS architectures are defined in terms of the agent roles according to coordination levels and the interaction among the agents [128] (See Fig.2.5). Agent interaction can be in the form of "message passing, requesting, negotiating or producing changes in their common environment" [16, p. 1048].

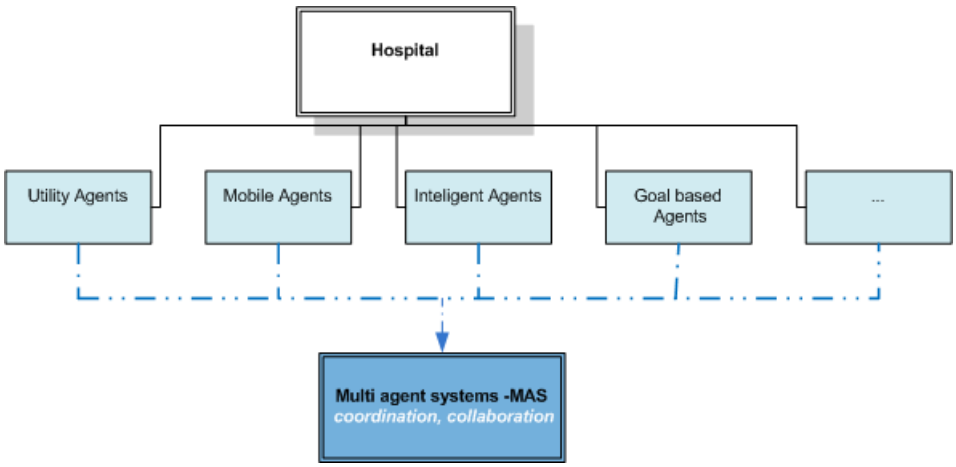

Figure 2.8: Types of Agents

According to Padghan and Winikopff [124] MAS agents in terms of structure should consist of:

- Actions, which represents the way the agent will react/interact within the environment

- Percepts, the ability to acquire relevant information
- Events, detecting a number of activities/changes/particular conditions
- Goals, follow objectives to be accomplished
- Plans, achieving goals
- Messages, way to communicate/interact with other agents
- Protocols, interaction rules

This research will include the use of software agents, particularly knowledge based agents in a MAS platform as a tool to be used to empower the sources of information to monitor data in real time. MAS have been mainly used in distributed environment applications given the characteristic to improve communication, coordination and support collaboration by working as a team rather than isolated units [35] [112].

As can be seen from [13][53][76][87][119] and [16][100], in BI agents and multi-agent systems have been used to respond pro-actively to determined tasks and to work in complex and dynamic environments. The proactive actions performed by an agent are frequently modelled in terms of goals.

The learning process and the ability to react to beliefs play an important role in the agents design and frameworks vary from the architecture level to the implementation level. There is no standard methodology to implement a multi-agent system. Nevertheless, some efforts have been proposed by industrial and research organizations such as FIPA and OMG [53][119].

2.6 BI in health care

Moving into the health care arena where the prototype of the framework will be implemented, patients, health care providers, payers and government demand more effective and efficient health care services. Thus, the health care industry needs innovative ways to re-invent core processes. The use of BI seems to be a possible solution to tackle this challenge.

The health care arena requires real time data management in areas such as patient monitoring, detection of adverse events, and adaptive responses to operational failures. Early

automated feedback prevents process breakdown and gives an early response when risk scenarios are present.

Effective and clear communication is vital to preserve patient safety in all health care activities. Decision making in this environment is typically complex and unstructured as clinical work is essentially interpretive, multitasking, collaborative, distributed, prudent and reactive [159][102].

In patient/individual disease monitoring a number of variables must be considered such as disease features, symptoms, and practitioner knowledge and skills. The use of a BI system in this scenario will include and facilitate the capture, storage, processing, communication, security and reporting of individual health care records [172].

Nowadays pervasive health care monitoring environments, as well as business activity monitoring environments, gather information from a variety of data sources. However, it includes new challenges because of the use of body and wireless sensors, non-traditional operational and transactional sources. This makes the health system more complex to monitor. Here data filtering and data fusion techniques using data warehouses, context awareness and knowledge generation using RFID, and data mining techniques to achieve reliability are some of the proposed approaches to achieve real time data monitoring for health care [141][159].

It can be seen that the health care enterprise has its basis in a complex and adaptive environment in which the field of real time decision support systems theory provides fertile ground for research on health care process improvement strategies [159]. However, to date BI research in healthcare has largely provided an overview of the advantages in knowledge gathering and data mining to support and increase the efficiency of decision making as a future challenge and possible solution to adopt.

Research presented in [172][102][156] addresses the need for using BI in health care but no tool or real implementation of a BI framework for e-health is proposed.

In the area of patient data mining a few tools have been used to provide new insights for process intervention. However, implementations are still very limited as the generation of false positives alerts is one of the problems and patient specific data processing in right time is still not achieved [156][50][93].

Intelligent diagnosis seems to be the area in which technology has demonstrated to be most effective as several applications have been developed to support disease diagnosis. This will be discussed in the following sections.

2.6.1 Clinical data warehouses

In health care, various components are connected and sometime overlapped [47]. Clinical data is stored in heterogeneous and distributed repositories across regional and international institutions around the globe [141]. Thus, moving into the specific health care domain, clinical data warehouses (CDW) can facilitate the analysis and access to data obtained in the patient care process which improves the quality of decision making and early process intervention[141].

Attributes of a CDW are given in [47][88][92][141] and [161]. CDW are more than a large collection of clinical data and generally data comes from other enterprise systems, devices and sensors with the data being commonly integrated into data marts according to the data warehouse architecture defined.

It seems that clinical applications require the construction of a more powerful data model than is usually provided by business approaches. According to [31, p. 8], "on average, patients might have hundreds of different facts describing their current situation". Complicated and unpredictable procedures require quick decisions. Thus, more advanced classification structures are necessary to provide continuous data monitoring and measuring.

In summary, it is necessary to manage large amounts of heterogeneous electronic heath records, (from general data to RX images) and also give efficient support to process query and analysis at any time (e.g. drug interactions, sensor measurements or laboratory tests). Electronic patient data travels from different organizations; such as public hospitals, general practitioners, private clinicians, government and insurance payers. Therefore, a robust architecture that enhances data sharing without duplication is required. It should provide the integration of information from heterogeneous sources, security access to patient's data (security infrastructure) and the access to information in a timely manner (real time).

Thus, timely response by on time data analysis to prevent patient deterioration and sound decision making, security infrastructure and effective data integration are some of the issues

still not fully addressed in the clinical data warehouse domain.

2.6.2 Patient disease monitoring

Medical diagnosis is highly variable and involves complex processes. It depends on decisions made by medical experts and to identify the nature and cause of diseases, it bases its judgement on medical knowledge (experience, reasoning, books and/or pathological tests) that comes from observed symptoms. Therefore, dealing with expert knowledge consideration, diagnosis systems have been inspired by the use of intelligent techniques to detect at an early stage patient symptoms which determine the signs of a particular disease.

A number of diagnosis systems have been developed [30][32][71][80][90][137]. These systems commonly involve active interaction among various medical knowledge modalities. To create intelligent decision support systems in the pervasive management of health care, multi-agent platforms, mobile agents, artificial neural networks and fuzzy logic are some of the major current techniques in use.

Fuzzy logic would probably be the most appropriate to use to simulate the uncertainty in decision-making (inexact data and knowledge). On the other side, neural networks are commonly used for prediction and classification problems due to their capability of modelling extremely complex and non-linear functions [32][129].

Applications for patient disease monitoring focus mainly in home care such as the research described in [52][36][5][22]. The main goal here is to control the disease processes by monitoring medical protocols and to help patients by maintaining their independence to a maximum level.

The use of new technology systems to support the existing procedures of medical services opens a vast potential for research and development. Nevertheless, current applications focus on diagnosis, or on site medical device monitoring rather than individual disease management.

2.6.3 Capturing and modelling medical knowledge

To analyse medical datasets, data mining has been successfully applied to several medical problem domains. For knowledge discovery in health data, neural networks, decision trees

and association rules are commonly used. For this research it is relevant to identify the main characteristics to consider in coronary heart disease patient. Medical knowledge in the heart disease arena has been obtained through the use of some of the techniques and methodologies discussed in the following paragraphs:

A combination of data mining techniques that include *support vector machines, artificial neural networks* and *decision trees,* were employed by Xing et.al [178] to predict a disease outcome (patient survival).

Neural network algorithms have been in use to cluster disease data. They have been demonstrated to be a powerful technique for the development of diagnosis systems through image processing [80]. In addition, *neuro fuzzy techniques* comprise the use of a fuzzy system placed before a neural network. This composite system can identify patterns. Context based patient case specific analysis, symptoms, patient's state (syndromes), and disease outcomes are identified with a satisfactory degree of certainty [30][32][127][178].

Another technique used to predict diseases is mining using *association rules.* These are combinatorial in nature being particularly useful to discover patterns. As a result, rules that are medically significant can be discovered to assist the practitioner in the task of diagnosing [121][137].

In systems in which there are a duality of objectives, subjective knowledge, automatic acquisition and system integration, the use of *case based reasoning* seems to be appropriate. Case based reasoning systems in medical diagnosis adapt old solutions to interpret new situations by using knowledge-based learning to reuse its own experience [71]. By reasoning through experience, medical data is mined first in the knowledge discovery phase using decision trees, association rules or others. Then rules are stored to be used for the diseases prognosis, to formalize the cases and to predict new cases [142].

Last but not least, *Mobile agents* in [22][35][100] are used to monitor, analyse, and manage the data of patients where abnormal patterns are detected to have an advance treatment and prevent loss of life. In these, mobile agents are focused on sensors that read patient data. Then as a team, mobile agents inform and keep a record of the health status of a patient by analysing current data to find abnormal patterns for which they have been trained.

It is necessary to highlight that in most of the diagnosis systems reviewed, systems give alerts based on historical data in which expert medical knowledge (a doctor) is used as basis of extracting relevant information and checking the accuracy of the alert [30][80][90]. Therefore, only a few approaches have been proposed as adaptive diagnosis systems [178]. Although those systems seem to be accurate in extracting data in real time, none of them achieve the challenge of integrating data from different data sources.

2.7 Summary of the research findings

Taking a bigger picture view across all the literature review conducted, it is possible to distinguish some problems that have not been yet resolved which could be considered in the proposed research.

Table 2.1 illustrates the main findings in the data warehouse area.

Data warehouses	
Research area	*Problem - Gap*
Data integration	** How to deal with partial information?*
RTDW	** How to distribute the Extraction, Transformation and Loading processes to best meet the need of real time decision making?* ** How to trigger real time system responses?* ** How to adapt to new business requirements?*

Table 2.1: Gap analysis- Data warehouses

Data warehouse technologies are used to facilitate data analysis from different data sources. Here data extraction, transformation and loading are essential components to build a data warehouse. ETL historically was considered a supporting task for the implementation of a data-warehouse. Nowadays it represents one of the open areas of research to achieve active monitoring and data fusion in real time. To our knowledge there is no ETL tool that has fully provided real time business intelligence as most of the current approaches focus on speeding the loading process and present scalability and consistency problems.

In BI choices of data generation flow solutions vary from batch versus stream, and from push versus pull. Currently in BI architectures all data to be analysed has to be consolidated in the warehouse first. So, queries are directed to the data sources to extract and integrate all the information [65]. These pull solutions have found federated systems the best way to

achieve data freshness in a timely manner. Nevertheless, real time alerting and reporting cannot be done in a query [20].

As can be seen, real time decision support using warehousing systems has at least four main areas of open research. The problem of how to deal with partial information will not be considered in this research. Therefore, it is assumed that data inconsistencies and duplications have been treated before data is been transferred to the central repository. This research will address three of the four problems identified which are:

- *How to distribute ETL to best meet the need of real time*: We propose an event driven approach as a way to sense and react in real time to certain environment conditions. Pre-filtering data by loading the data in small batches (trickle feed and table partitions) seems to be more efficient than consolidating all the information available. Nevertheless, we believe that data content is absolutely relevant to monitor processes and therefore, data can be processed and analysed in the sources, by enabling learning capabilities on them. In this manner, local knowledge is used as rules to determine when and how to react when unusual data content has been captured at any source of information.

  Thus, sources of information could react by pushing data into the warehouse when there is not enough knowledge to take a decision and when data content has changed. In a monitoring mechanism once updates have been detected at the sources, the content of the data changed will be analysed to look for data abnormalities or an unusual behaviour in terms of the type of data received, and if the data received is within the accepted ranges of value. Consequently, only unusual data changes will be transferred to the data warehouse. Real time response will be kept to a minimum latency by eliminating the data availability gap which will enable organizations to concentrate on accessing and processing valuable data only (data content changes).

- *How to trigger real time alerts*: Most of the current approaches analyse data once it has been consolidated as only their data content is checked. In our approach, if the monitoring mechanism and data analysis is moved to the sources, an alerts mechanism can be triggered as soon as data anomalies are found. Data sources will have intelligent

capabilities to learn and monitor data content according to predefined rules.

- *Adaptation to new business requirements*: A data warehouse architecture capable of learning from the environment (disease) will be designed as the way to automatically update the events that trigger the systems alarms (patient's medical conditions, current data values). Therefore, this design will enable logical adaptability of the data warehouse content which is adaptation of the knowledge rather than data warehouse structure.

In the monitoring area the research problems found were:

Active monitoring	
Research area	*Problem - Gap*
Real time Monitoring	** How to support "real time" analysis, reporting and detection of adverse events?* ** How to effectively apply intelligence to data?* ** How to update that intelligence?* ** How to perform adaptive responses to breakdown in normal processes?*

Table 2.2: Gap analysis - Active monitoring

How to support real time analysis, reporting and detection of adverse events: Even though real time cannot be achieved because as soon as a data entry is perceived or viewed and analysed data is no longer real time, by distributing the decision making among all the actors involved in the process, and empowering the data sources to intelligently pre-filter data, data analysis starts from the very beginning. These actions could then effectively help to decrease the latency for alerting and reporting when risk scenarios are present.

In the proposed research real time will be taken as follows:

- Any valuable data (key components to monitor in a patient) that changes in any of the the sources of information will trigger a certain event (analyse/alert/update/transference) to save time in the decision making process before the deadline is reached (clearly a serious risk for a patient).

By monitoring these *key components*, evaluating them, and then responding to them on time the proposed architecture is able to respond or alert to unusual/risk scenarios faster than traditional data warehousing strategies have allowed.

It may be possible to enable local autonomy at the level of the data sources to push information (transfer data) to the warehouse, rather than pulling or scheduling the data extraction as most current approaches do. Thus, as soon as valuable data arrives at the data source, it will be analysed based on previous knowledge (historical information) otherwise it will be immediately sent to the warehouse. To enable data push, agents can be used as these are defined as entities that enable local intelligence to react in particular environments. Through this way, the use of local knowledge in distributed environments can be employed to effectively alert as soon as abnormal data behaviours have been detected in the pre-analysis stage executed at any source.

The general goal of the research proposed is to deal with the main two factors that add uncertainty and risks to RDSS which are computational latency (CL) *by not being able to make a decision because of delayed status reports* and, synchronization latency (SL)*by taking decisions based on reports generated with out of date data* [179]. Most of the efforts to tackle CL and SL are related to defining information value from the business side and query processing for the technological side. Research in the SL domain at the moment only shows experimental solutions [23][24][136] [146] [109].

The gap of *How to effectively apply intelligence to data, knowledge updates and adaptive response to breakdown process* for active monitoring will be addressed by providing intelligent capabilities to determine the triggers that will analyse/push the data to feed the central repository and/or to alert the decision makers responsible to monitor the data.

Intelligent data analyses will be performed in two levels:

- Firstly to detect data abnormalities (unusual data content).
- Secondly, to detect when a data update is necessary as the base of knowledge as changed and therefore the rules to monitor must be updated.

Health care is one of the scenarios in which immediate response must be achieved in order to prevent tragic outcomes. Individual patient disease management in real time has not been addressed and therefore, it has been chosen as the area to implement the prototype of the real time data management mechanism described here.

Chapter 3

Methodology

This chapter describes and explains the methodology and methods deployed to undertake and answer the research questions of this study.

3.1 Introduction

Software engineering research in Information Systems has its basis in the use and development of new technologies/concepts to improve the effectiveness and/or efficiency of processes to fulfil business needs [59][43]. Information systems research also pursues the development and communication of knowledge concerning the areas of information technology management and the use of information technology for organizational and managerial purposes [69].

Thus software engineering research as a component of information systems has proposed new methods and artefacts such as algorithms, interfaces, frameworks and methodologies. These artefacts are created to help to understand, improve and/or explain different aspects of information systems [174][173][19]. Thus, by designing technological artefacts when producing scientific knowledge, a connection between knowledge and practice has been created.

Marcos [99] identified three main research areas in Software engineering/Information systems which depending on the field, software engineering science, software science, and the science of information systems, they might have a different character and approach to follow. These are described in Table 3.1.

Hence, having a look at this project's research question; *Does the local empowerment of*

Science	Object of study	Character	Methods
Software Engineering science	*Construction of new objects*	Engineering	Qualitative creative
Software science	Object constructed	Empirical	Quantitative
Science of Information systems	*Implementation and uses of objects constructed*	Cultural and social	Qualitative

Table 3.1: Software engineering, problems, sciences and methods [99]

data sources to pre-analyse and push data to feed a data warehouse enable better response and alerting mechanisms to detect risk scenarios in real time?, it can be seen that this research has an engineering character and overlaps the sciences of software engineering and information systems as it creates a new object/artefact. This new artefact is designed to be used as a way to improve decision making in business activity monitoring as well as making a more efficient and effective architecture to monitor a variety of data sources to feed a data warehouse in real time.

Thus, the current research overlaps the areas of Software engineering and Information systems science because:

- It pursues an improvement in the way to gather and analyse data to monitor in real time, which touches the basis of action research as it endeavours to solve a real world problem and at the same time studies the experience of solving the problem [38],by creating a framework to monitor data in real time. It aims to contribute with a new way to manage and consolidate data in distributed environments *Therefore, it creates a new object, a new way to do things*

- It aims to be more efficient to monitor data under unusual scenarios in distributed environments and be more effective to support decision making. Therefore, a prototype will be implemented to prove and validate the concepts proposed. *In this way the object will be implemented and its use will be studied in the field of health informatics.*

While software engineering mainly focuses on all types of software it can be seen that information systems research, by definition, focusses on information systems software such as data support and decision support systems instead [106]. Therefore, even though this research overlaps both fields it will mainly focuses on Information systems research.

In the following section a brief discussion of the methodologies used for artefacts design are presented to finish in section 3.3 with the methodology used in this dissertation.

3.2 Artefact development methodologies

Traditional sciences, sociological and biological for example, have a good understanding of the research strategies to follow. They usually deal with the study of existing phenomena or objects, while engineering sciences deal with the study of methods and the necessary techniques for the creation of new objects such as methods and techniques [99][147].

Wieringa, and Heerkens [174] showed that Software Engineering (SE) and Information systems (IS) discussions related to the contribution of design and empirical research are similar. While "IS complains about a lack of relevance in the results produced by rigorous empirical methods, SE complains about the lack of relevance of design results and attempts to increase relevance by promoting rigorous empirical methods" [174, p. 1]. It seems that design and empirical research by itself do not guarantee relevance, this is why the relationship between both is being proposed by the use of methodologies such as design research and the implementation of formal design processes as a whole while constructing objects/artefacts [97] such as software architectures in software engineering research. IS research theorizes about the development of new artefacts but also pursues the justification behind these theories [97].

In some empirical studies research seems to be disconnected from practice, thus design, also called development research, aims to generate knowledge by testing how theory might work when applied to practice [138]. Most design research activities include investigation of a problem, the solution for the problem which is usually an artefact and where the properties of a proposed solution are established, and the evaluation of the implementation where the main findings of the solution are discussed [175].

There are two main research methodologies to be used in the development of artefacts. These can be seen in Fig.3.1

One of the first research methodologies proposed for artefacts development in information systems was presented by Nunamaker [115]. This research approach focused on describing the development research processes involved in systems development research. Systems development research has its basis in that the the new concept proposed will contribute to the creation of new knowledge in the area of application and the purpose of studying it is relevant to the domain. A system developed by itself is not research if it does not contribute knowl-

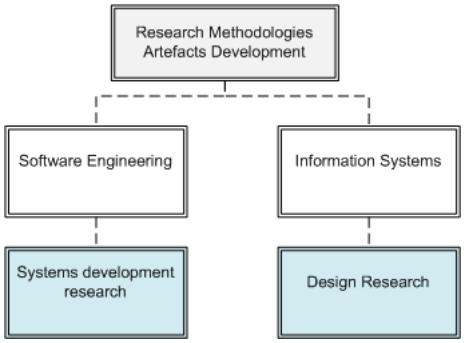

Figure 3.1: Main research methodologies involved in artefacts development

edge to the domain [18][60]. Thus, the developed system will serve as a proof of concept for the fundamental research but it will also provide an artefact that eventually be the focus of continuing and future research [115].

It has been considered relevant to classify systems development research as part of software engineering (SE) as SE provides the methods and tools for which this research can be validated (prototyping and walkthrough methods for example).

Systems development research processes involve [115]:

- *The construct of a conceptual framework*, where the research questions, the systems requirements and the study of the relevant areas comprised in the research are examined and described.

- *The development of a system architecture*, where the architecture design and the functionalities of its components are proposed.

- The analysis and design of the system, where the processes involved in the systems functions are designed along with an evaluation of the possible alternatives to develop the system.

- The construction of a system prototype, where the concepts and frameworks design insights, the complexity and problems of the system are learnt.

- The observation and evaluation of the systems, where generally by a case study the system is tested and new theories and models based on this evaluation are proposed.

If the developing of a new system or the improvement of an existing one provides a better solution to existing information system problems, can be tested, and makes a real contribution to the domain in study, then the development can be considered as research.

Moving on to development of artefacts in information systems, lately design research has been attracting significant attention as a research methodology. In general design can be considered as the activity that rules the structure, function and development of a system [98]. It correlates knowledge and practice by producing scientific knowledge by designing artefacts [173][68][167].

Design research has been described as a research that involves "the analysis and performance of designed artefacts to understand, explain and very frequently improve the behaviour of aspects of information systems [167]". It generally includes the following research processes [167]:

- The awareness of the problem, where the proposal for a new research effort is obtained given the identification of research gap in a determined domain.
- The suggestion phase, where a tentative design for the artefact is obtained.
- The development, where the artefact is implemented. The deepness of the implementation will vary depending on the artefact. An algorithm might require just a formal proof while a software architecture might need a software prototype.
- The evaluation, where the artefact constructed is evaluated to certain criteria which is generally tied to the description of the research made in the awareness phase (proposal).
- The conclusion, where the future research and main research findings are summarized.

It can be seen that there exist in general at least four main research phases to undertake in artefacts design , activities that are connected to the two previous domains described [48] [98][117] [167][68]:

- Understanding the problem: This involves the gathering of information necessary to understand the domain in which the research will be conducted. It sets the theory

behind the domain, the gaps and helps to understand the problem and refine the research question.

- Requirements and objectives: This phase describes the requirements to design the artefact, it sets the expected functionalities and the main objectives involved in the new way of doing the things. It might also be required in this phase to consider the objectives and the main requirements of the artefact to fulfil practice needs.

- Find the solution for the problem (design): In this phase the design model is established as an architecture, thus the design is used as a vehicle for analysis during the implementation and/or validation phase.

- Validation and Evaluation of the solution: Here the validation and evaluation of the solution proposed is contrasted with the research question to determine whether the artefact fulfils the outcomes determined and helps to answer the research question established in the first phase.

Systems development can be considered as a part of artefact design in terms of the development of architectures and software systems artefacts. In these terms the research output will be a new method or framework, in other words a set of steps to perform a task in a specific domain of information systems. Consequently, design science research seems to be more appropriate to be used as the core methodology to be applied in this dissertation. However some methods and activities of systems development will also be considered. This will be described in the following section.

3.3 Research methodology phases

It can be seen that the five main processes of design research can be complemented by the systems development activities in the following way (See Fig. 3.2).

1. Phase 1: *Understanding the problem* To analyse the problem, gaps and the theory behind the areas involved in this research a literature review was conducted. Through the development of the literature review presented in Chapter 2 of the current document,

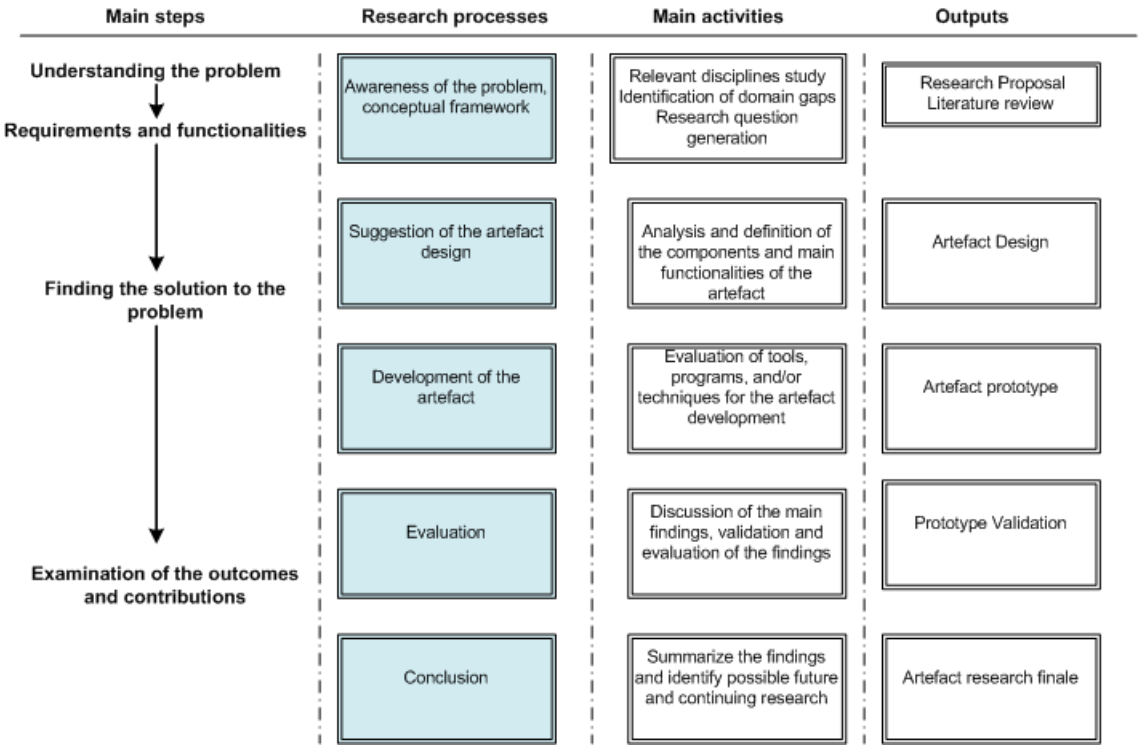

Figure 3.2: Research process in design research adapted from [167] and [68]

an understanding of the subject area to conceptualize the research problem and body of knowledge was obtained.

2. Phase 2: *Requirements and objectives*

   After the problem was identified a set of objectives and requirements were obtained and gathered to suggest a tentative design for the framework proposed. To do this the literature review was used along with information taken from meetings and interviews with business activity monitoring technicians and health care specialists. Chapter 4 shows the main requirements and the model, set of steps required and involved in the TTL architecture design, obtained as a result of these activities.

3. Phase 3: *Find the solution for the problem*

   Given this research proposes a new way to design a monitoring infrastructure to feed a data warehouse in real time, the development of a prototype was needed to demonstrate the validity of the design [59]. Thus, the main part of the architecture proposed was

built by creating a prototype that monitors coronary heart disease patients in real time. Chapter 5 shows the main findings of the prototype implementation.

4. Phase 4: *Validation and Evaluation of the solution*

   To validate and as a way to examine the outcomes of the design proposed a walkthrough was used to evaluate the prototype [149]. Through this a review of the findings was obtained in which specialists (decision makers) tested the prototype in different scenarios and gave us their opinion in four main aspects of effectiveness.

   The four main characteristics to evaluate the prototype in terms of effectiveness, quality, and some other ways of improving performance of health systems by using e-health were obtained from the research conducted by Arah et al. [7]

   An evaluation in terms of computational efficiency was also obtained. This was done by comparing and contrasting the design proposed in this research with the deployment of a traditional loading mechanism to feed a data warehouse. Thus, an analytical evaluation and a comparison of both architectures, a traditional approach and the TTL approach, in terms of computational performance by experimentation was obtained.

   The validation and verification in terms of efficiency and effectiveness of the framework proposed in this dissertation is presented and discussed in chapter 6.

Chapter 4

TTL: A Transformation, Transference and Loading Approach for Active Monitoring

4.1 Introduction

Most BI architectures use centralized decision making support architectures where commonly a data warehouse technology is used as the way to consolidate, analyse and report data to support decision making based on all the information that has been collected from a variety of data sources. In Data Warehouse (DW) environments, operational processes move data from a variety of sources to the warehouse. This movement of data is usually driven from the central repository, the warehouse, which is programmed to request (query) data from the sources in a timely manner. Data refreshments are then time dependant.

Traditional data warehouses are refreshed in a periodic manner, usually on a daily basis (off-peak hours), where the operational sources and the data warehouse experience low load conditions. Therefore, there is a cooling-off period between business transactions and their representation in the data warehouse, with the most recent data unavailable for analysis as it is caught in the operational sources. Thus, traditional DW systems might impose unacceptable delays due to their batch nature [77].

Even though data warehouses are fully recognized in organizations as the middle-ware layer between applications and decision support systems, given its batch nature for today's requirements of real time information integration and analysis there is still the need to come up with the right architecture. Requirements have changed and real time transactions are now demanded to support operational decision making in real time [9][144].

Organizations are aiming to use information technology solutions to capture time-critical operational processes in the shortest time possible. Intelligence, active responses and adaptiveness are some of the main characteristics expected for business intelligence solutions to perform on the fly data analysis [64][9][103].

In the following paragraphs a description of the architecture proposed to design a real time data warehouse architecture to monitor data in real time is presented. A view of how the data sources must be prepared and empowered to be able to capture meaningful events, the local pre-analysis component, and the response mechanism necessary to alert to unusual behaviours will be explained.

4.2 Event driven decision making processes with data warehouses

From a business perspective, there is a need to collect meaningful data as soon as possible to improve and decrease the time of decision making with the use of BI solutions. Hackathorn's chart [64] gives a perspective of the relationship between information value and delivery time (latency) in right time reporting environments (See Fig.4.1).

The lapse of time between the time a business event is captured and the point it is moved into the data warehouse to be delivered to the right user is called latency. In general organizations should cope with at least three different processes that create latency before data is analysed. Preceding the time that data is captured data latency is present, after data is consolidated there is analysis latency and just before an action is taken by the user decision latency exists. The higher the time to deliver information, the less value the event captured will have for the organization.

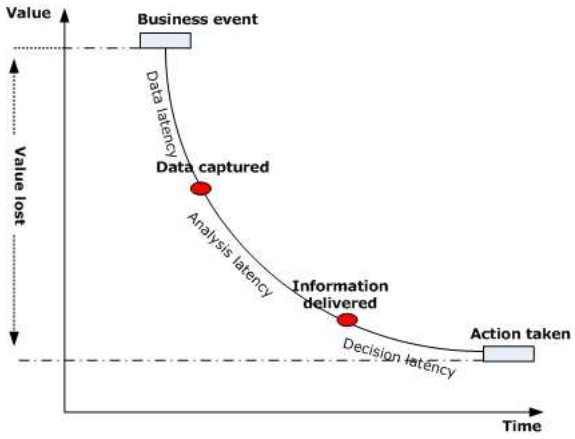

Figure 4.1: Information value vs delivery time by [64]

One of the relevant comments of Hackathorn [64] in terms of using BI solutions to add value to information, is that he highlights the idea of performing data analysis in a different way by capturing data and analysing it before it is delivered to the right user. Commonly business activity monitoring architectures do not do this as it is necessary to perform several activities before data is captured. The order of these can be seen in See Fig.4.2.

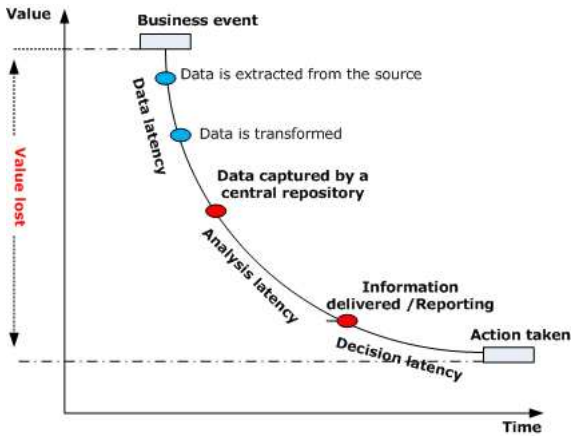

Figure 4.2: ETL in right time, variation of [64]

Data needs to be extracted, cleaned and transformed before being loaded to the warehouse. The warehouse performs the analysis and information is delivered and reported to the right

user.

Business activity monitoring (BAM) architectures focus on monitoring and capturing meaningful events at the sources in real time to provide decision makers with up to date information. BAM "real time" designs use approaches such as event driven architectures mainly based on event stream processing techniques to enable events to be analysed without being first transformed and stored in "a database", and pull architectures or data warehouses mainly based on change data capture, message queues or data base triggers to update the data more frequently and to consolidate data from a variety of sources into a communal repository.

Real time BAM architectures show a data latency typically in the range of minutes to hours. The analysis of the data is usually manual and depends on the report mechanism associated to the tool and data base manager expertise [41] [44]. Semantic errors or data content errors are mainly detected, if detected at all, once data is being analysed (after data consolidation). Therefore, the decision latency is still significant and the data provided for reporting to decision makers might not be fully accurate.

Even though fast refreshment for data warehousing is made possible with the use of incremental techniques such as change data capture, the role of the DBA is closely coupled to the efficiency, performance and reliability of the BI environment. Most BI solutions monitor data as a solo rather than a continuous series of events and do not have associated automatic notification when opportunities occur and/or problems arise. Therefore, companies might not have the real-time component associated with their BI systems [51]. Diaz [41] argues that real time data warehousing does not enable event driven decision making and therefore it does not provide real time performance as *pure BAM systems* are fundamentally different from business performance management and traditional ETL systems.

The main considerations and challenges to use real time data warehouses in BAM according to [41] can be summarized as follows:

- Loading a large amount of data in the warehouse can considerably affect the response to events mainly because reports can only be generated after data has been loaded.

- Data models in the warehouse are not easily modified. To display changes in a model implies a time of re-indexing and re-loading usually of weeks or months.

- Correlation of actions or events over time are not supported. The need to eliminate false positives given a solo event does not seem to be meaningful until a number of other events have occurred (time series correlation)

Diaz [41] highlighted the challenge of achieving data quality in BAM architectures by detecting errors on time, a challenge that could be approached by providing reliable sources of cleansed data, in which data errors will be detected before the data is loaded in the warehouse.

Thus, it can be seen that there is still an open area of research in the development of an architecture that can effectively monitor unusual behaviours. By monitoring just valuable/meaningful data and identifying abnormal behaviours, it is possible to react only once changes to normal (accepted) parameters are detected [44][26].

Because ETL techniques are difficult to scale up to address the challenge of data loading, performance and low latency to provide real-time decision support using data warehouses, a new approach is presented here. The framework introduces an approach for designing real-time data warehouse architectures to support Business Activity Monitoring under unusual behaviours in which traditional ETL does not apply. A event driven approach with the use of a Transformation, Transference and Loading (TTL) mechanism is proposed instead.

An event driven approach is used as a way to sense and react in real time to changes in environment conditions. Data is pre-filtered, processed and analysed at the sources by enabling learning capabilities in them. Thus, sources of information sense and react by then *pushing*, rather than "pulling", data into the warehouse only if needed. This gives the possibility to pre-filter, analyse and transform data at the source levels. Real time response is kept to a minimum latency by eliminating the data availability gap to perform data analysis which enables organizations to concentrate on accessing and processing valuable/meaningful data only.

By aiming to minimize data analysis latency and therefore increasing the value of information in organizations, the sequence of activities to be performed in the TTL approach could be described in accordance with Hackathorn's model as follows (See Figure 4.3):

To deal with latency issues the TTL architecture empowers the data sources with intel-

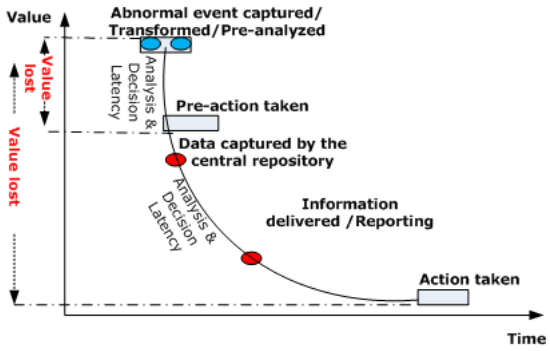

Figure 4.3: TTL in right time, variation of [64]

ligent capabilities to monitor and pre-analyse valuable data. The pre-analysis is performed by using a Multi Agent System architecture (MAS). This paradigm has become more and more important in many aspects of computer science by introducing the issues of distributed intelligence and interaction. Agents learn and reveal the data activity patterns through day to day measurements and the data history contained in each source of information. An agent reacts, after the pre-analysis has been done, by sending alarms according to changes in those patterns [25], or transfers data to the warehouse because more actions are needed, or if there is not enough knowledge to perform an action at the source level.

A partial view of the TTL approach can be seen in Fig.4.4. The architecture has four levels to achieve the real time monitoring mechanism. These levels have been organized in a different way to traditional ETL processes in order to start pre-analysing data and alerting from the very beginning, as soon as data has been captured at the source level.

The three main processes of the TTL framework can be summarized as:

- Pre-analysis process, where meaningful events are captured and pre-analysed before being loaded to the warehouse.

- Knowledge updates, here a mechanism to learn or to take the knowledge stored in historical data and from day to day events is implemented. These will allow the architecture to be adaptive. The architecture can be logically adapted in the sense of updating the knowledge/rules for which actions for a unusual events and/or data behaviour are taken,

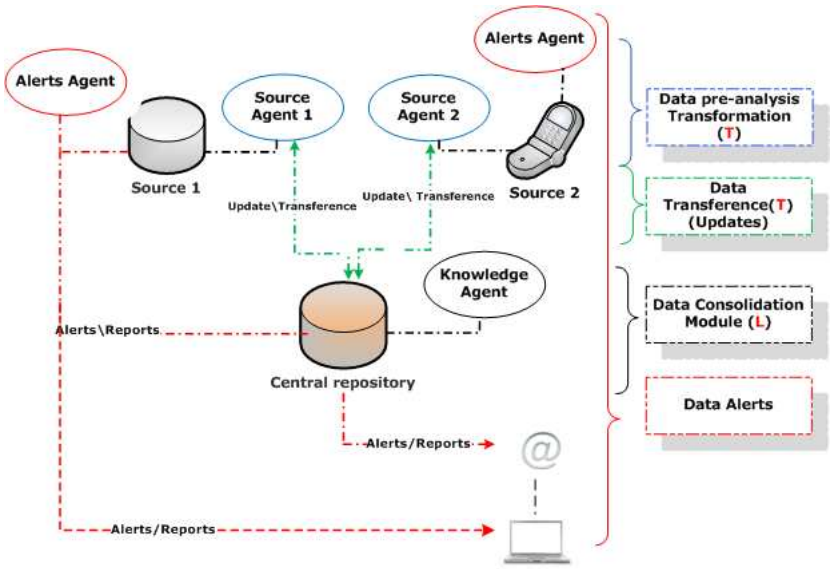

Figure 4.4: The TTL approach

and how events (knowledge rules) should be pre-analysed and captured.

- Real time alerts and responses, where the outcomes and responses to unusual data/events and alerting mechanisms to decision makers and users are processed.

These processes, the architecture levels, and the multi agent infrastructure needed to achieve real time response are explained in the following sections.

4.3 Identifying and modelling the knowledge

4.3.1 Identification of data - organizational requirements

There are certain aspects to consider before modelling the TTL real time architecture:

Firstly, the TTL architecture works with the idea of monitoring meaningful data in abnormal/unusual scenarios for some entities of the data source only. So, first of all the organizations have to decide what will be the sector to monitor, the entities to monitor, and which are going to be the meaningful values to monitor for them (See Fig.4.5). The sector provides the boundaries and the general knowledge for the monitoring mechanism.

Secondly, the monitoring mechanism is based on alerting and reacting when unusual behaviours are present in the data. Abnormal behaviours can be described as a particular event or as a series of data events that do not correspond to or fit in accepted data ranges. As an example, a unusual behaviour could be a change in the number of sales in a particular day when monitoring a product, or changes in blood pressure data ranges when monitoring a patient. These changes are detected given the knowledge extracted from historical data which in this example might be able to determine and establish the level of acceptance of blood pressure for a particular patient, or to give a prediction about how much of a product will be sold daily.

Thus, the architecture monitors only a subset of the data that resides at the data sources. That subset, any relevant entity to monitor at the source level, is being called a *compound* for this research. A compound, a subset of data, is the object continuously monitored and for which changes in data will be pre-analysed.

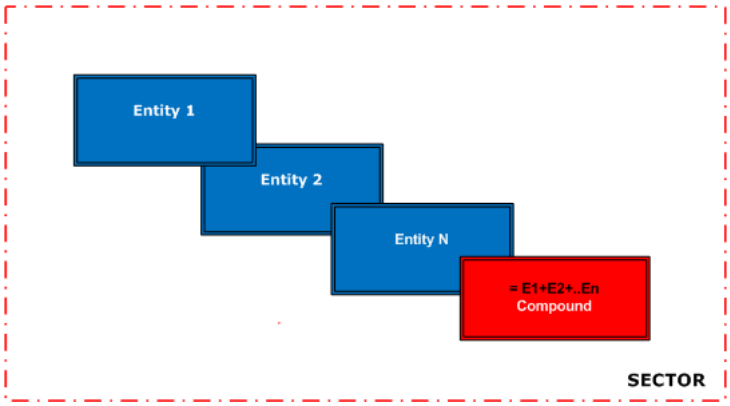

Figure 4.5: Relation of sector vs compound

The knowledge of the sector that the compound belongs to gives the boundaries for the monitoring mechanism. In a sector, organizational knowledge about standards, policies, metrics and others is extracted and used as general rules for the monitoring mechanism. The sector knowledge must govern compound rules.

As previously mentioned a compound is pre-analysed given the knowledge that can be extracted from all the historical data of it that exists at the sources. A compound's knowledge

gives an idea of how the data of an entity usually behaves and what the accepted ranges and boundaries for it are. The compound's knowledge provides the compound with *particular rules* to the monitoring mechanism such as what the data should look like for an specific entity, which are the ranges for usual readings, or which are the data ranges to tolerate or to decide when it is behaving normally.

A number of entities, such as a number of patients, products and accounts, form a compound. However, patient normal ranges (blood pressure ranges and weight for example) differ from patient to patient. A product sales and characteristics differ from product to product. Bank accounts activities differ from customer to customer too. Nevertheless, banks, health care practitioners and business organizations set the rules in which all these different activities should be monitored and which should be the urgency to have responses and alerts in scenarios of unusual behaviours. Through this way entities that behave differently and have their own particular rules, but belong to the same sector are monitored.

A sector is where a compound exists. A sector can be a particular company or industry and therefore a compound for that sector can be any entity (individual, process, or product) that is relevant to monitor. A sector and a compound depending on the organization could be for example:

Sector 1 *Hospital*:

- Compound to monitor *"Patients with a particular disease"* as show in Fig.4.6

Sector 2 *Banking Industry*:

- Compound to monitor *"Certain client transactions such as withdrawals"* as shown in Fig.4.7.

Once the sector and its entities have been identified, it is necessary to specify the ownership of the data. That means to specify where the entities's data resides (data bases, systems files, etc.) and how this information is connected with the different systems (other data bases and areas) of the sector. That allows for the identification of sources of data, operational users and decision makers and the discovery of possible duplications and/or interconnections of business information.

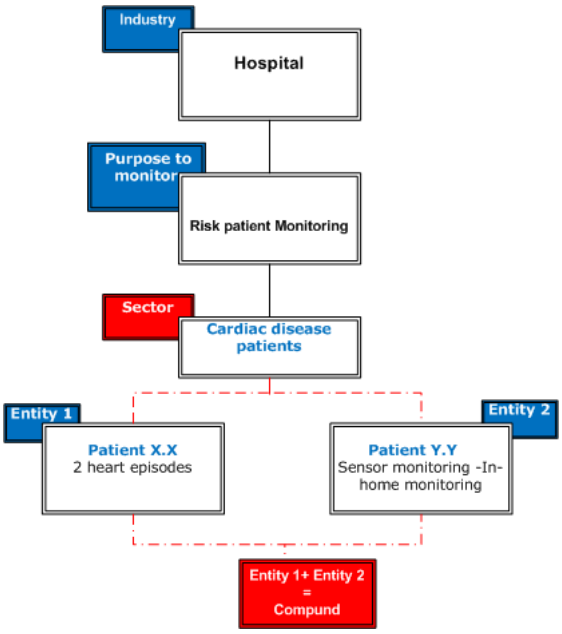

Figure 4.6: Health sector- Risk patients monitoring

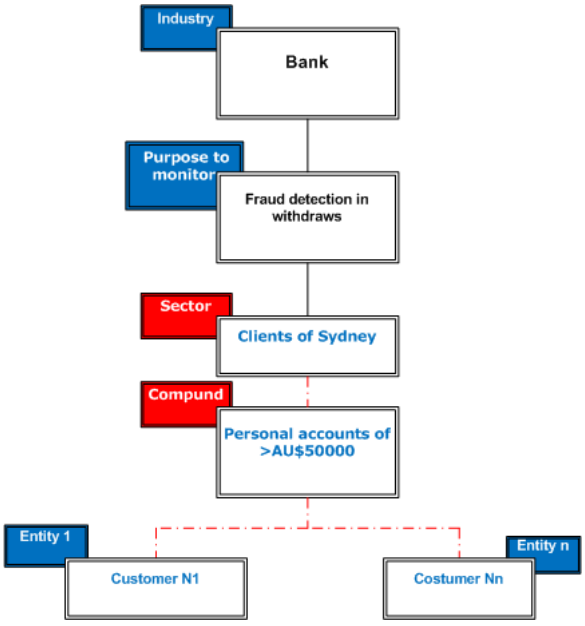

Figure 4.7: Banks sector - Fraud detection

Therefore, the organization should define requirements such as:

- *Identify the sector* that will undergo the monitoring and have a full description and understanding (knowledge) of the environment/boundaries in which the sector should be monitored - What sector needs to be monitored?
- *Define the entities* of a compound. Define what will be the subset of data in each source to monitor. - Which data is relevant to monitor for the sector and therefore needs to be closely overseen?
- *Define the sector's base of knowledge* - What are the rules for the sector?
- *Define the compound's rules and its priority* - What are the accepted rules of each entity?
- *Identify ownership of entities* - Sources of data and users - Where does the data to process reside? Who needs the information and who needs to be notified of unusual behaviours to perform the decision making in real time?
- *Define alerting mechanisms*- how and when a user will be notified of unusual behaviours?

When defining compound rules and their priority, it will be also necessary to define the series of events that together imply an alert and represent an unusual behaviour.

Combination of symptoms for a patient in a certain period of time could trigger a health episode (cardiac arrest for example) or change in withdrawal locations (different countries for example) for the same customer might mean fraud detection. Therefore, the *Identification of possible compound events* or episodes in a certain period of time is also necessary. These episodes must come from the general knowledge and boundaries of a sector.

A good way to identify the knowledge of a sector to determine episodes over time, would be to consider expert panel discussions. Episodes can be then taken from the experience of managers, health care practitioners and any decision maker that given the knowledge of the sector that they have could help determine what could be an episode.

Computational algorithms, such as mining techniques, can also be used to identify events, trends and patterns at the source data for each entity to monitor. Several mining mechanisms

and pattern matching methodologies can be applied to detect trends and data patterns in historical data [74] [153][71]. Which one to use will depend on the sector, organization and the type of data to monitor. By working with only current information the TTL architecture might never be able to detect trends and long-term patterns of behaviour, therefore historical information is important for understanding the seasonality of business operations, in which the data evidences regular and predictable changes to which each compound is exposed.

These events and patterns for each entity and the rules of the sector are the incoming knowledge for monitoring in each data source. This knowledge allows the empowering of the data sources to contrast normal patterns of data with the current data that is arriving at a source and to also detect a time series of events that imply an episode of a particular entity. These will be discussed in depth in the pre-analysis tier of the TTL architecture (Section.4.4.1).

All the organizational requirements described in this section are critical to successfully implementing the monitoring mechanism that the TTL architecture will use. However, they are out of the technical boundaries of the architecture itself.

4.3.2 Preparation of data - technical requirements

Once the sector, its entities, rules, data sources and users have been identified preparation of data is needed.

Orr [123, p. 67] states that "no serious information system has data quality of 100%. Data quality means that the quality of the data in information systems is accurate enough, timely enough, and consistent enough for the organization to survive and make reasonable decisions". The problem with this statement is that real data changes continuously and that the data that is stored at the sources is static. Thus, systems that aim to capture the real world, must contemplate a " mechanism to synchronize the data in the system with changes in the real world and therefore feedback is necessary" [123, p. 67].

In the data warehouse environment data entry and capture is prone to errors. To correct these exploration of data sets is needed. Within this context data cleaning is a crucial part of the ETL processes and it can be described as the computational processes used to eliminate

errors and inconsistencies in data [135] [96]. There are several tools such as [95] and [45] to utilize to perform integrity analysis and to locate data errors in data sets. Moreover, nowadays relational data integrity to check data consistency, including entity referential, and column integrity, can be achieved using relational data base queries in many database systems such as Oracle and MySql.

In data capture from sensor devices, missed readings because of low power hardware and loss of wireless communication for example, are frequent. Besides, unreliable readings are often present because individual sensor readings are " imprecise or unreliable" [79, p. 83].

A discussion of automated detection mechanisms for detecting errors in data sets and in sensor devices for pervasive environments is presented in [96] and [79] respectively. A set of methods to use such as pattern recognition and association rules were evaluated to address the problem of automatic identification of errors in data sets. A framework for evaluating data quality tools is proposed by Thion-Goasdou et.al [165].

Error detection in data sets has been studied for some time and the TTL architecture assumes the use of a data cleansing mechanism to achieve data quality problems while capturing and transferring data to the central repository.

Nevertheless, once a data source is identified, the logical structure of the data contained there, and its relationship with other sources, must be specified. This process allows the establishment of data content (data types), logical structure, and the knowledge to recognize data inconsistencies and/or errors once data is captured. This process of error detection is developed as part of the TTL architecture. Only data cleansing itself will not be considered at this stage. As an example a view of some of the types of data used is provided in figure 4.8.

The identification of data structures, patterns and logic of the data make it possible for the TTL architecture to detect data inconsistencies that are not visible to a schema level and cannot be solved using traditional cleansing tools. These inconsistencies can be described as: missing data in a not null field (right structure and type but dummy value detected), contradictory and erroneous data (structure and type correct but contradictory to the range of data e.g. if the weight captured is 71kg when the normal range of weight should be

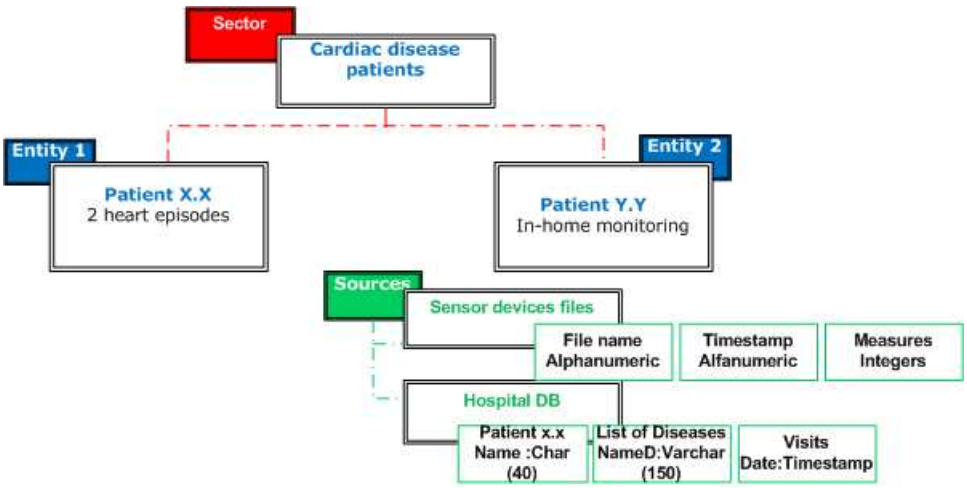

Figure 4.8: Health sector - entities, sources and data types

30kg), and embedded values problems (wrong fields insertion). The theory behind these types of inconsistencies are taken from [10]. The error detection conducted to detect errors at the sources of data as part of the pre-analysis will be explained in the pre-analysis section (Section 4.4.1).

4.3.3 Achieving safety while monitoring Individual Sensitive Data

Individuals Sensitive Data (ISD), such as patients or bank clients, should not be processed without their consent. Taking the example of the UK Data Protection Act of 1198 [125, p. F124,7A] personal health data cannot be processed in the absence of explicit consent unless they are needed for medical purposes or undertaken by a professional who in the circumstances owes the duty of confidentiality. Thus, the organization must ensure consumers/clients data security while monitoring their data. The disclosure of ISD can be regulated through client consent [33].

The use of consent mechanisms is recommended while processing ISD which enhances privacy and confidentiality because it contains the specific conditions under which certain data can be retrieved and processed. E-consents vary from a single full disclosure mechanism to applications in which system's access is performed according to user roles and consumer

security profiles.

It is also important to consider the use of anonymized data and pseudonomization techniques as methods to protect ISD when data is extracted from different data sources. To minimize *data breach* the proposed architecture focuses on processing ISD at the local source (owner of information) first before sending data (transference) to the consolidated repository. In this way each source of information will know what data is being analysed and when it is being transferred, then when it needs to be sent, anonymization techniques must be in use before transmission. As there is currently no complete standard for pseudonomization, this implementation will be considered as future research and not part of the TTL framework.

In the following sections the way the architecture pre-analyzes data and the main constraints to achieve the recognition of abnormal behaviours in real time are described.

4.4 Modelling of Transference, Transformation and Loading of data

This section presents the structure of the four main levels of architecture proposed. In here the architecture design constraints and main considerations are discussed.

To describe the TTL architecture's structure and main processes a UML approach was followed. For each module of the proposed design a number of use cases and/or sequence diagrams will be provided.

4.4.1 Pre-analysis and transformation tier

To provide integrated access to a variety of heterogeneous sources such as data bases and sensor devices, the pre-analysis and transference module is responsible for monitoring data compounds (E1..En) in each source, and for transferring data to the central repository whenever needed. Here the mechanisms for pre-analysing and pushing the data need to be established (See Fig.4.9).

The object responsible for monitoring all the meaningful data contained in each source is called Source Agent. To do this:

Figure 4.9: Pre-analysis/Transference tier

- *Source Agent* is subscribed to the entity's ID to monitor in each source of information.
- *Source Agent* through a set of specific rules, taken from the base of knowledge (historical data) of the sector and compound, and with the use of an event condition mechanism, compares entity accepted parameters with the data changes captured at each source to take the decision to perform an action. This action can be information to be delivered to the central repository, to send an alert to data managers, or not doing anything as the data captured (even though is relevant to monitor) is within the normal range of parameter.

The main activities performed in this module can be seen in Fig.4.10.

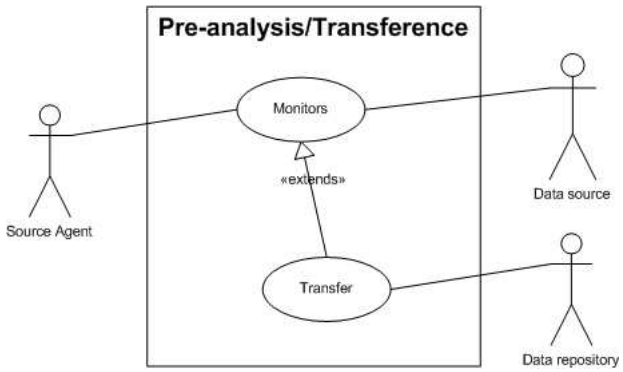

Figure 4.10: Use case: Pre-analysis/Transference

Source Agent monitors an entity by comparing and contrasting its knowledge, rules of the entity and sector rules, with the data/events that are being captured at the sources.

Source Agent does not monitor all data available in each source. It monitors only the relevant data for that entity that has been declared important to monitor (i.e particular patient, client or product). Therefore, this is an *individuals* specific framework in which local knowledge has been taken from a subset of the original data called compound and from the sector where the compound belongs.

The mechanism for comparing and contrasting data at the sources, and to therefore empower the sources to pre-analyze data, is based on the Event Condition Action paradigm (ECA) [177] [169]. ECA rules have three main components:

- The Event component: An event will trigger the evaluation of the ECA rule. There could be also the possibility to have a time series of events, or episodes, that will be checked out to act or to move the system to a next state. Van Bemmel [169] gives a good example of ECA rules in health care for a diagnostic system in which a complex pattern of patient's symptoms or monitored signs might trigger the treatment phase, next stage, or activities such as the reservation of a surgery room and the scheduling of the surgeon.

- The Condition part: It represents the way in which the actions of the rule are implemented given the occurrence of events. It is usual to describe conditions as predicates or boolean expressions.

- The Action part: Contains the list of functions and/or operations of the rule(s) that gives the behaviour of the ECA/system.

Nowadays, most database management systems have integrated a mechanism to use ECA rules. Thus, a source is active and empowered to advise the source agent that data/events related to a particular entity have changed. Then, source agent will be able to activate its own rules of pre-analysis to take specific actions, such as alerting decision makers or transferring data to the central repository for further analysis or storage. An example of a SQL ECA rule is given as follows:

The rules of source agents are the patterns and accepted ranges extracted from the sector, *general knowledge* and the historical data of each entity *compound's knowledge* to monitor.

Algorithm 1 ECA rules general example

```
1: WHEN entity.atribute EQUALS "something" AND entity.atribute EQUALS "something"
2: FOLLOWED-BY entity.atribute EQUALS "something"
3: WITHIN "some time"
4: ACTION "alert"
```

The general base of knowledge is the area of interest to monitor, i.e. the sector's knowledge. These could be rules, policies, metrics, and/or standards that are applied in the sector and will affect the way a compound should behave.

In the case of a bank organization, the sector's knowledge could be the policy related to how much money clients are allowed to take in each withdrawal. In a health sector, general knowledge could be represented for the parameters and general guidelines to follow to assess a patient with heart disease.

Compound knowledge is composed of all the accepted ranges and patterns taken from the entities's historical data. These could be the accepted ranges of each symptom or features of a cardiac disease patient such as the accepted ranges of blood pressure and weight. Compound's knowledge differs from entity to entity as in the case of the health care sector in which patients have different values of weight, height and blood pressure.

Thus, as soon as valuable data arrives at a source of information a trigger alerts and sends the valuable data/or event to *SA* which based on condition-action rules (if-then) performs a check that includes assessing data structure, data relevance and data content. An example of this is given as follows:

Algorithm 2 Source agent - pre-analysing data

```
1: if the data changed (dc) in the source is meaningful data to monitor then
2:     check it against data.structure.type accepted for the compound
3:     if dc.type!= data.structure.type.accepted then
4:         react/alert
5:     end if
6: else
7:     check dc against accepted ranges
8:     if dc = accepted.range.feature then
9:         go to sleep because no action is needed
10:    else
11:        reacts/alert/transform/transference
12:    end if
13: end if
```

Transformation is one of the main stages and time consuming activities while integrating data from heterogeneous data sources. It is needed as to standardize the data structures and the way data is obtained and stored across different organizations data sources might be impossible to achieve. Different data sources could have different keys for representing the same entity, entities could contain unnecessary attributes and/or fields that are irrelevant or missing that might affect the information integrity.

For the TTL architecture, Source agent should only perform a check of data structures and types based on accepted data patterns taken from the entities historical knowledge. Thus, as part of the pre-analysis logic data errors can be detected by knowing the structure of the data and the accepted content of it. Consequently, source agent knows how the data content and structure should look.

Data might be transformed on the fly at the sources when the dc's type is accepted but the content of dc is not within the accepted ranges. The transformation technique to be used, such as conversion, filtering, sorting and translating, will always depend on the data to be integrated from the sources. Some sources might need more or less transformation work than others and therefore no specific recommendation is given here.

Data is transferred to the central repository when there is not enough knowledge to take action at the source level and a consolidated view of data between the general knowledge and possibly other compounds data is needed. A view of these will be explained in the following section.

4.4.2 Data transference/updates tier

We argue that usually there is enough knowledge already in operational systems to be extracted and used to empower the sources of information at a local level to monitor meaningful data changes. By achieving a certain level of autonomy and intelligence employing agents, a push system with local empowerment in a distributed architecture is deployed.

Data is transferred to the central repository once atypical behaviours have been pre-analysed and a full view of the situation must be seen for another agent given source agent knowledge is not enough to take an action at the source level. The pre-analysis gives the

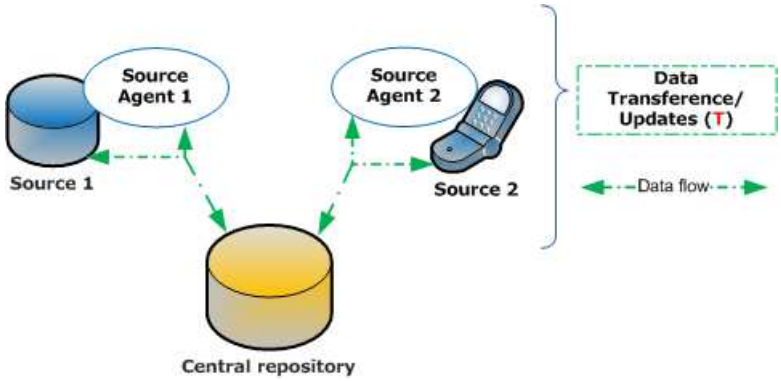

Figure 4.11: Data Transference and Updates Tier

opportunity to take decisions and to alert as soon as abnormal behaviours have been detected and before all relevant data is consolidated in the central repository.

As seen in Fig.4.11., the flow of data is bidirectional in this module too. It will be possible to achieve logical data warehouse adaptation if *Source Agents* are available to learn from the changes made at the source level. Repetitive changes in behaviours might mean an update of the knowledge or a change of requirements for the compound monitored. These will allow the architecture to have a mechanism to determine and react against new requirements and knowledge compound changes providing logical adaptability for the architecture.

There are several learning mechanisms that could be used to automatically update the knowledge of Source Agent such as statistical reasoning, fuzzy logic and decision trees [34][32]. Which one to use will depend on the sector domains, type of data and monitoring objective. A semi automatic update mechanism can also be possible by simply counting data changes in time series (days, weeks or months). This could be implemented with expert knowledge feedback, thus as the user (expert) is notified of continuous changes and as soon as it is accepted as a true change the rule of source agent to monitor that compound can be updated (See Fig.4.12).

Compound knowledge is historical information that has been processed and included into the Source Agent reasoning mechanism and made available for active use in the decision process. This knowledge can be updated according to different scenarios to give Source agent

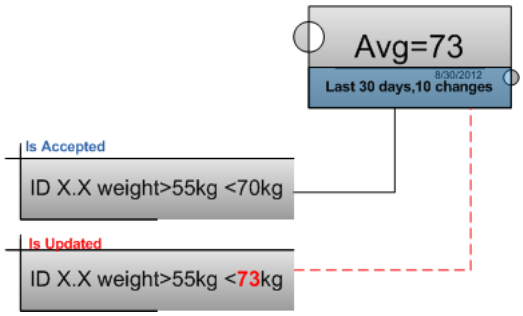

Figure 4.12: Data updates in compound knowledge

the capability to automatically build up learning models to be used to answer what-if questions and to respond to particular and different compound events. Each entity (a particular patient) that belongs to a compound (patient with coronary heart disease for example) might have different information to be extracted.

Therefore, the sector provides the general rules to monitor but each entity that belongs to the compound might have different rules to be monitored. Source agent knowledge is then conformed by general rules (sector rules) but also by individual rules which are the particular rules that each entity that belongs to the compound has.

4.4.3 Data alerts tier

Once data has been analysed, whether this analysis is performed in the central repository or at the source level, an action is taken and alerts are sent to the decision makers because an unusual/abnormal behaviour in the content data for a particular entity has been detected (See Fig.4.13).

The main activities performed in the data alerts tier are checking the alert type and according to that the alert is driven and sent to the determined user. These activities are displayed in Fig.4.14.

A classification of events, data behaviours, and level of importance of changes is needed to be implemented to provide customized alerts. In a monitoring mechanism the more customized the alert the better to provide the right information to the right user at the right

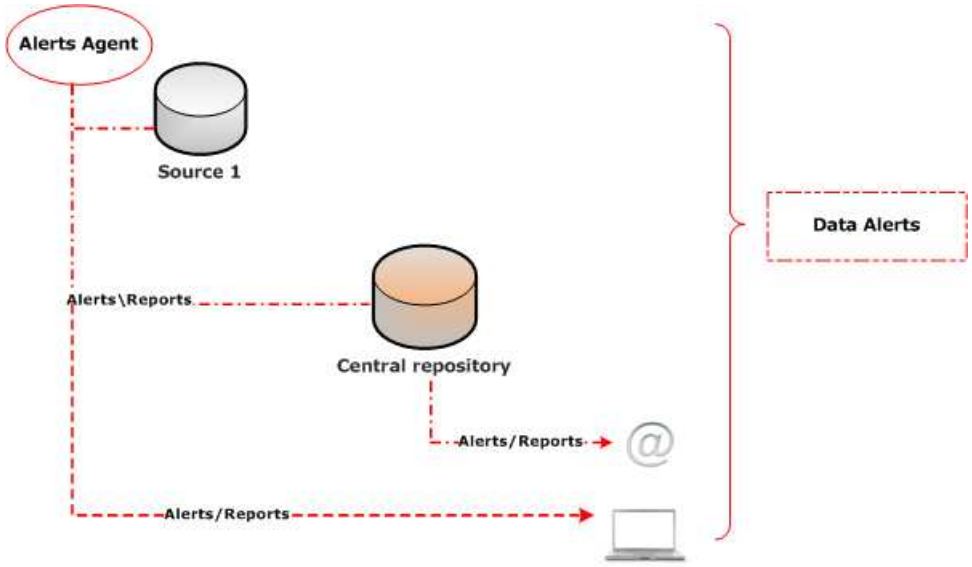

Figure 4.13: Data Alerts tier

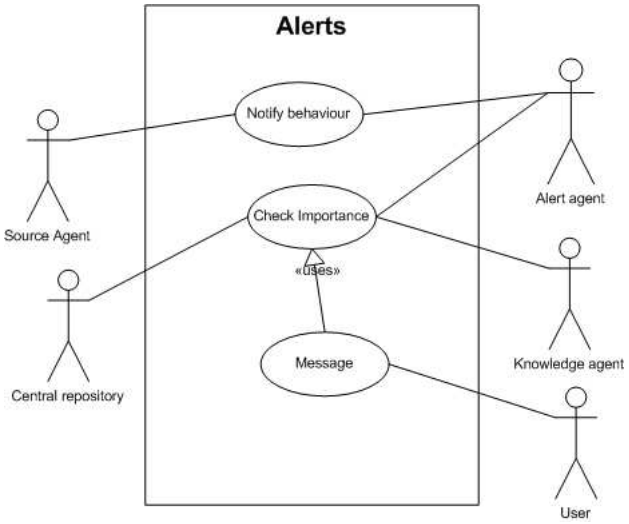

Figure 4.14: Use case: Alerts

time.

In the health care sector, monitoring alerts mechanisms could be different for each user, patient and health care practitioners for example, depending on the type of change. Consequently, an alert could be sent to a patient given the sensor device that it uses may not be

Algorithm 3 Source agent - pre-analysing data

```
1: if behaviour =high.importance then
2:     notify.decision.maker.high & save.behaviour
3: end if
4: if behaviour = medium.importance then
5:     notify.decision.maker.medium & save.behaviour
6: end if
7: if behaviour = data.error then
8:     notify.user.source
9: end if
10: if behaviour = unknown.behaviour then
11:     notify.user.source
12: end if
```

working properly, due to the fact that the data pre-analysed by the source agent seems to be incomplete or is not within accepted ranges. Moreover, alerts could be also sent to the decision makers given the behaviours represent a risk for the entity to monitor, a risk to a patient in this case.

The classification of risks and the level of importance of behaviours might be taken from the general and compound knowledge and the classification of users needs to be performed in the organizational requirements stage.

4.4.4 Data consolidation tier

The main activities of the data consolidation tier includes data transference, transformation and loading (Figure 4.15).

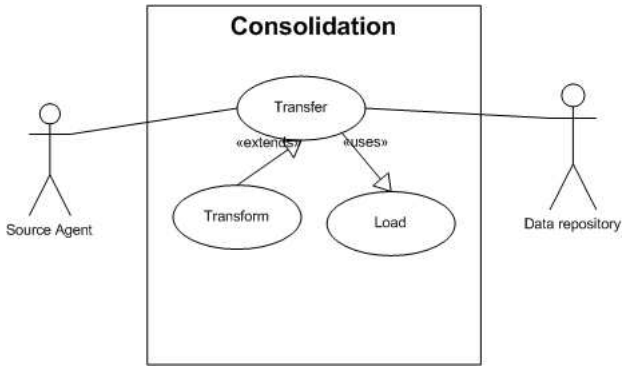

Figure 4.15: Use case: Consolidation

The consolidation repository, a data warehouse, has a view of all the entity events pushed by the sources in a determined period of time and that have been pre-analysed by Source Agent. The *Knowledge Agent* resides here. If the set of data being monitored at the sources does not match any of the rules that *Source Agent* has, the information is then compared with the full base of knowledge of the compound that resides in *Knowledge agent*.

Knowledge agent can also request more information from a different *Source agent*, an agent in another data source, to have a full view of the events and data situation before performing an action.

The central repository, with the use of a Knowledge agent, supports the implementation of local knowledge updates, the consolidation of compound's meaningful events and acts as a communication mechanism among agents.

A discussion about the multi agent system platform that supports the Transference, Transformation and Loading architecture is given in the following sections.

4.5 The multi-agent system for TTL

Software agent characteristics such as autonomy and the ability to exhibit some intelligent behaviours to achieve different goals makes the use of them feasible and relevant to deal with distributed systems. Multi-agent system architectures nowadays play a valuable role in the area of distributed artificial intelligence.

This section provides an overview of the Multi-agent system employed as part of the TTL framework. The main types of agents used, their characteristics and interaction mechanisms are described bellow.

4.5.1 Agent types

In agent programming frameworks *a goal* can be described as a core functionality for which the agent will make plans and follow actions. In data monitoring, different monitoring goals might require different actions and therefore different information to be communicated [82].

For the framework presented here, the type of agent goals used in the TTL architecture were defined from Ferguson [51], whose research discussed the main characteristics that orga-

nizations should obtain while using technology to create multiple intelligent agents in business activity monitoring.

According to Ferguson [51] real time monitoring technologies should allow companies to:

- listen and capture specific events as a business process begins
- integrate information to make possible automated analysis as soon as each event being monitored occurs to produce intelligence, and
- capture and use rules to know whether it is necessary to take an action according to predefined rules or use of intelligence.

Following this description, it is possible to identify four main goals. Listening and capturing events, integrating information, achieving intelligent data analysis, and alerting whenever it is needed. In order to accomplish these goals the following agents are used:

- A Facilitator Agent: Which facilitates the communication among other agents and mediates to take decisions about unusual behaviour of entities when information at the source level is incomplete. This agent coordinates the compound knowledge updates of source agents and as [2] states, other agents should give up their autonomy to the facilitator services. This facilitator agent has knowledge about the sector, knowledge about the processes that are performed in the sector and can extract knowledge from the entities, request more information from other monitoring agents, before taking a decision. Knowledge agent decisions range from loading data to the warehouse, integrating events, validating the information and actions based on the monitoring agent findings and informing possible outcomes to the communicator agent.
  - The goals of the Facilitator Agent are to integrate information in the central repository and to achieve intelligent data analysis. The facilitator agent for the TTL framework is *Knowledge Agent*.
- A Monitoring Agent: Which monitors entities and detects unusual patterns and data behaviours at the source level. This agent also filters data from the sources to be

integrated in the warehouse and sets suggestions based on the findings. A monitoring agent has knowledge about the data content of a particular entity to monitor, the accepted parameters of the sector and controls the events that occur in time series based in the rules extracted from the historical information of a compound.

  - The goals of the monitoring agent are to listen and capture events, transform data in case it needs to be transferred to knowledge agent, respond to unusual events, and achieve intelligent pre-data analysis at the source level. The monitoring agent of the TTL architecture is *Source agent* and there could be as many source agents at data sources as the monitoring sector needs to consolidate information.

- A Communicator Agent: Which coordinates the alerting and response mechanisms of the facilitator agent and monitors agent findings. A communicator agent has knowledge about the users (decision makers and operational users) and the way messages must be delivered to them.

  - The goals of the communicator agent are to deliver the information and alert responses to the right users of the platform. The TTL communicator agent is *Alert agent*.

4.5.2 Agent Communication

Given each agent has the ability to communicate with other agents based on the outcome of the findings, it was not considered necessary to have a centralized super-agent as part of the TTL framework. It can be seen that the TTL framework relays in distributed mediators instead (source agent and knowledge agent) to handle the agent cooperation and communication among agents. If more information is needed to perform an action at the source level Source Agent will interact with the Knowledge Agent. Both Knowledge agent and Source agent can communicate with Alert agent to deliver the findings and alerts to users.

There is no standardized definition of the way agents should interact with each other [168]. Agent communications languages vary from platform to platform. Thus, the way TTL agents communication mechanism will be implemented depends on the agent's architecture

tool selected. This does not form a relevant part of the architecture design and will not be discussed here.

As a way to explain the main components to consider while implementing a communication mechanism among agents a general description of agent communication is given as follows.

There are three main components in multi-agent interaction described in [2]. These are:

- A common communication language such as Knowledge query manipulation language (KQML) or the Foundation for Intelligent Physical Agents agent communication language (FIPA ACL). Both rely on speech act theory and are declarative communication languages.
- A common content of communication such as the Knowledge Interchange Format (KNI).
- A common ontology. An ontology is described as a specification schema which describes the concepts/rules and their relationship in a specific domain. Agents should rely on an ontology to interpret communication interactions.

4.5.3 Agents Interaction

The way agents will interact relies on the communication mechanism used. An overview of the sequence of activities that should be considered as the way agents will interact with each other for source agent, knowledge agent and alert agent can be seen as follows (See Fig.4.16).

The data sources, the source agents, the knowledge agent, the alert agent and the warehouse are the main objects of the diagram. It can be seen that the flow of activities goes mainly to the warehouse. The warehouse does not query for data, agents push information and the flow of events to it.

Information is captured, cleaned and pre-analysed at the local level by source agent. Source agent pushes pre-analysed data and events to the warehouse when source data is relevant but presenting an abnormal behaviour. Source agent can also send the findings to knowledge agent because further analysis of the data that has changed at the sources is needed. This is done because knowledge agent can request more information from other sources. Source agent might send the findings to alert agents when the pre-analysis is enough

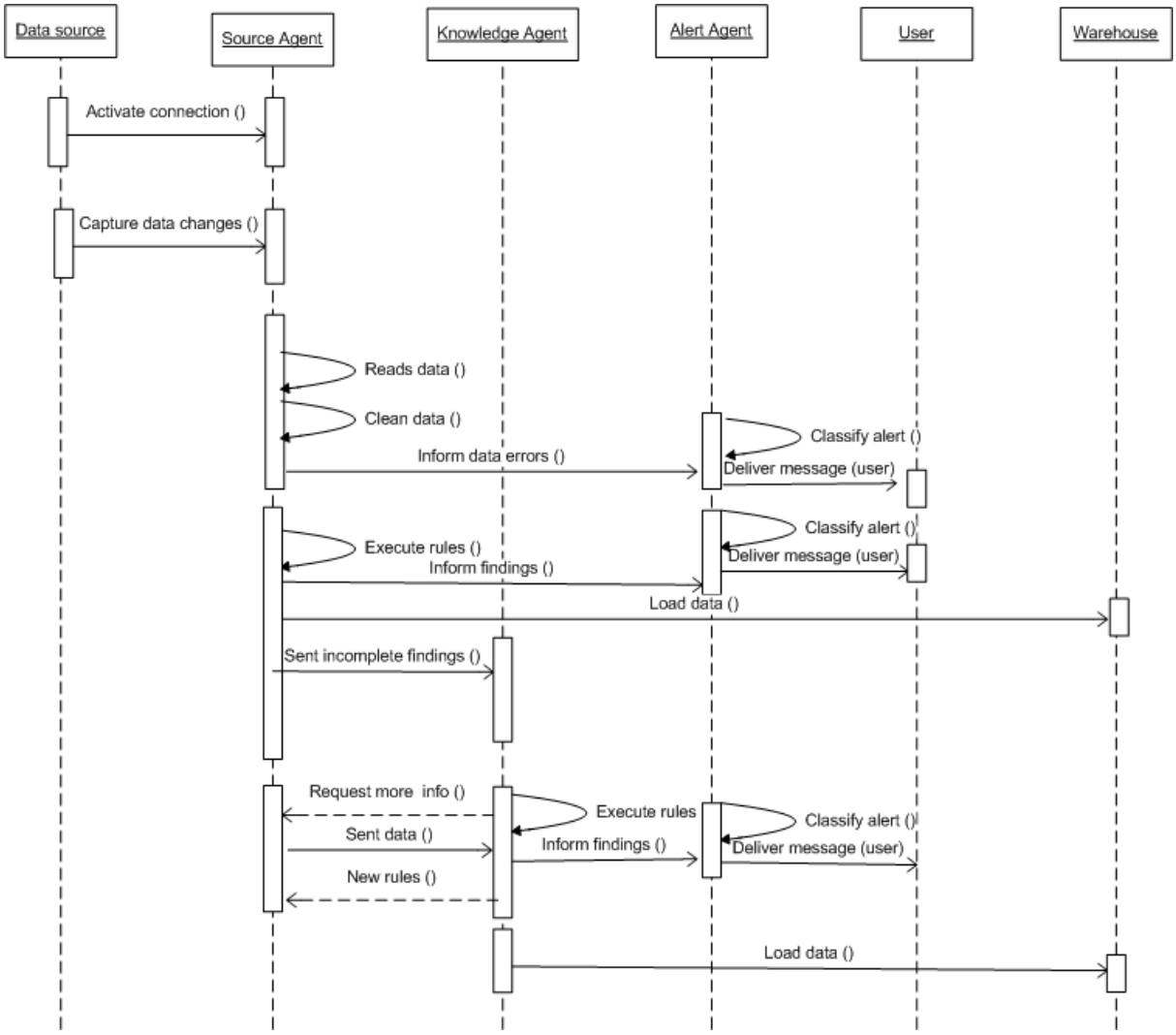

Figure 4.16: Agents activities sequence diagram

to perform an alert to a pre-determined user, operational user or decision maker (patient and/or health care staff for example), because unusual behaviours and/or data errors have been detected.

Knowledge agents can request more information from other sources agents (agents that rely on other sources of data) to complete the data analysis. Knowledge agent loads the pre-analysed data into the warehouse and/or sends the findings to alert agent to notify decision makers of unusual behaviours.

Alert agents on the other side read and classify the findings of source agents and knowledge agents according to user needs. In this way messages are delivered to the right users only.

4.6 TTL architecture discussion

For the research being described here real time has been used as follows:

Any valuable data (key features to monitor at the sources level) that changes will trigger and determine certain reactions (analyse/ alert/ update/ transference) in order to save time in the decision making process before the organization deadline is reached.

A deadline is reached once the time accepted to react and to know that the compound being monitored at the source level is showing a unusual behaviour is attained.

Rather than providing a traditional BI architecture in which a centralized mechanism is implemented to start with the data analysis necessary, the TTL architecture splits the data analysis in two main areas to distribute the decision making in two units. *The local decision making unit* enables data pre-filtering and alerting as soon as relevant data to monitor has been detected from the sources of information and pushes data to the warehouse for further storage and/or decision making, and the centralized *Knowledge management unit* on the other side drives the updating mechanism for the knowledge that the local decision making area uses and helps to go further with the data analysis whenever that is needed (See Fig. 4.17)

The *local decision making* unit will enable organizations to use current domain knowledge to take decisions as soon as data changes become available in the entity being monitored given that unusual behaviours or data abnormalities have been detected. The *centralized knowledge management unit* will enable the organizations to have a flexible architecture in terms of having technology able to cope with frequently changing business environments as it allows knowledge updates and feedback to the local decision unit.

Consequently, the TTL architecture then follows the main characteristics that Hackathorn [64] proposed for a real time monitoring system in terms of providing a framework that is *intelligent*, *active*, and *adaptive* to support right time and sound decision making. Intelligence to detect unusual data patterns at a local level as soon as data changes have been captured

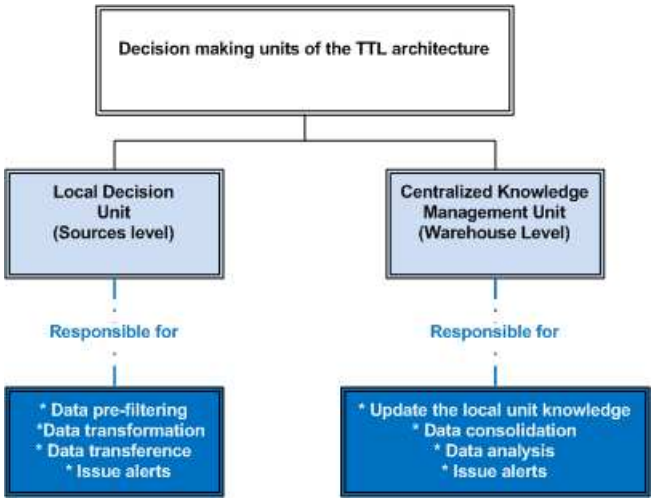

Figure 4.17: TTL levels

in a source. Active in the sense of working autonomously to pre-analyse data and perform alerts and adaptive to be prepared to learn from findings to update the rules that drive the monitoring processes.

By monitoring *key behaviours*, evaluating and then responding to abnormalities only the proposed architecture should be able to respond and/or alert to risk scenarios in a more effective way than traditional right time data warehousing strategies. A discussion of these will be presented in the following validation and verification chapters.

Chapter 5

TTL Implementation

To validate the architecture proposed a prototype for an active data monitoring architecture was developed. The area in which the prototype will work and monitor data was selected to be health care. For the prototype the main effort was to come up with a technology that could enable the empowering of the sources of information to use domain knowledge to perform local decision making to react, monitor, and feed a data warehouse in real time once relevant data to monitor has changed and been detected. In this chapter an overview of the prototype implementation is provided.

Section one introduces the application domain. Section two describes the prototype implementation of the local decision unit, it explains how the pre-analysis, transference, and alerting mechanisms were implemented. The third section summarizes the main findings of the implementation.

5.1 The application domain

The effectiveness of the framework proposed has been assessed by testing it in the area of health informatics. A solution to manage coronary disease patients has been proposed by designing an architecture that:

- learns and reveals heart disease activity patterns through day to day measurements and the clinical history of an individual patient,

- reacts in real time by sending alarms according to changes in those patterns to health care staff,
- and can be upgraded to be adaptive to new system conditions and changes in health care requirements.

Coronary disease monitoring of patients takes place at each step of patient management, from disease detection to disease prognosis and from surgery to recovery. During routine screening and day to day measurements such as sensor devices, patient knowledge can be obtained and therefore risk scenarios can be detected.

Why health care and coronary disease monitoring? Health information is widely distributed and involves many different processes and interactions. It contains individual patient information collected and stored by a person (a health professional for example) or a combination of many sets of individual information collected by more than one entity (hospitals, private institutions, health funds and others) which is stored in different locations in large databases.

The nature of health data makes its management complex. At high levels health information is used for policy, strategic planning, research and education. However, at low levels, it can be used for individual health services such as treatments and prescriptions. Therefore, one of the main problems in this area is data management in the sense of being able to access and use heterogeneous data (different data structures and data types for example) to provide on time care and rapid response to clinical decisions. Clinical data warehouses (CDW) in this context can facilitate the analysis and access to data obtained in the patient care process to improve the quality of decision making and to help with patient data consolidation by assisting with the extraction of only that data which is valuable for patient monitoring [86].

Nowadays *pervasive health care* monitoring environments, in the same way as in business, gather data from a variety of sources, but they include new challenges because of the use of body and wireless sensors which makes the system more complex to monitor in real time. The use of BI tools are still very limited in healthcare given the generation of false positive alerts which delays patient specific data processing in right time for clinical decision making [32] [50].

Traditional clinical data warehousing architectures utilize one or more on-line transaction processing (OLTP) databases before data is moved into a data warehouse on a monthly, weekly, or daily basis. Usually the patient data is processed in a staging file before being added to the data warehouse. In general patient data must be loaded regularly in order to facilitate the monitoring process. To perform this operation, data from various operational systems need to be extracted and copied into the warehouse (See Fig.5.1).

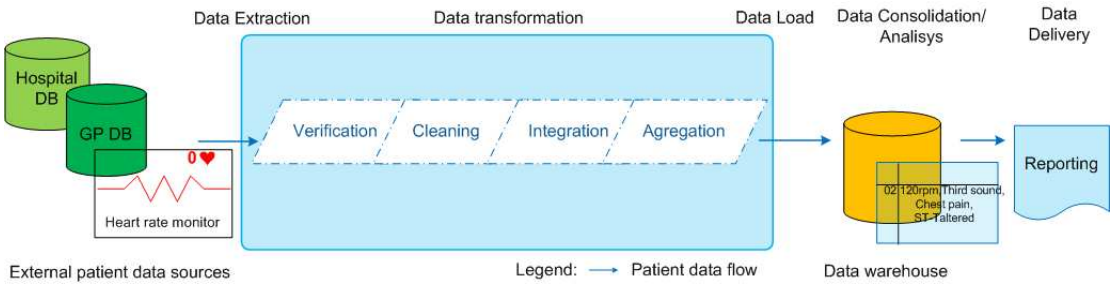

Figure 5.1: Data warehouse in healthcare

As can be seen in Fig.5.1, before reaching the reporting stage data needs to pass through at least four main stages data extraction, data transformation which includes; data verification, data cleaning, data integration and aggregation, then data is consolidated in a warehouse for further analysis and data reporting. Medical data bases store the data of thousands of patients that needs to be extracted before it can be analysed. Therefore, ETL needs to be programmed to consolidate all patient data to then report main findings to the health care staff. Taking the case of a patient that is presenting an episode of myocardial infarction and is using a monitoring system designed under current CDW technologies, the process of monitoring in a traditional data warehouse system could be as follows:

- Patient reports chest pain
- GP keeps a record of this symptom in the DB.
- GP detects third heart sound during examination
- GP orders a ST-T Scan

- GP keeps records of the examination and visit
- Patient is sent to Pathology
- ST-T is taken
- ST-T is Informed (analysed by a specialist)
- ST-T report is sent to the GP via system
- GP is informed of patient results
- Patient comes back to the GP / GP calls the patient as ST-T results are back
- GP checks patient history
- GP orders a procedure
- Patient is safe (hopefully)

If the patient data is consolidated, and the General Practitioner (GP) data base and the pathology data base are connected, in the best case scenario the episode will be controlled in a day or two. This will depend on GP availability, person and systems response (data analysis and reporting). In this scenario, data is not kept to learn from the patient disease prognosis and there are additional factors, like systems latency because of data extraction transformation and loading and person's availability, that will affect a rapid response and patient outcomes.

E-health data management is characterized by high pressure and timely access. People (i.e. doctors and nurses), organizations (i.e. administrators and health insurance groups) and systems such as referral and billing need simultaneous access to information. Physicians need information immediately in order to make vital decisions. Therefore, long system response time due to a high data load cannot be tolerated.

For the example previously provided, if the data extraction is performed daily the system may not provide information in time to notify that a patient is at risk which will leave all the responsibility to the health care staff availability and expertise to go back and check the report

test for the patient from the system. Real time decision support technologies enable doctors, patients and the health care staff to respond to a crisis and prevent problems by providing early indications of deterioration in the patient. Consequently, the use of data monitoring mechanisms, by using the patient information collected from any source, can provide timely advice to the health care staff when patient risk episodes are present.

As can be seen, health care decisions by nature are mostly critical and time related. Due to these factors, the ongoing monitoring of heart disease patient's data area was selected to implement the prototype.

5.1.1 Identifying the base of knowledge for heart disease monitoring

According to Chizner, [29] the diagnosis of heart disease can be made on the basis of the five finger approach. This includes a careful detailed history, electrocardiogram and laboratory tests, physical examination and a chest x-ray. In addition to that, the most common symptoms and features of patients with heart disease include chest pain and discomfort, breath shortness and palpitations among others.

Diagnosis of heart disease is then often made by looking at the patient history along with the experience and expertise of the health care staff, which some times includes the past experience of the practitioner in treating a patient with similar conditions. However, in particular cardiac disease diagnosis practitioners might be challenged as individuals might present some typical symptoms, some will show atypical symptoms while others will manifest none at all.

Identifying the important features of the disease while ignoring those which confuse the diagnosis may smooth the progress of physicians to make prompt diagnosis and treatment decisions. The research of Hongmei et al. [70] identified the main clinical features to consider for heart disease diagnosis. Using a realcoded genetic algorithm, a nature inspired search algorithm, 40 clinical features of heart disease were reduced to a list of 24. This new list represents the critical diagnostic features for 5 major heart diseases and should be considered to be the most relevant for diagnosis.

To build the prototype in the health care arena, a panel of 4 experts (Cardiologists) was

formed with the idea of using expert knowledge to identify the sector and the relevant features to monitor and for which alerts needed to be sent. The sector was defined to be coronary heart disease patient monitoring, and after discussions and using the list of 24 features created by Hongmei et al. [70] a sub list of 15 symptoms relevant to monitor in ongoing coronary heart disease patient assessment was obtained.

It is necessary to highlight that the medical panel decided to remain anonymous so no responsibility for the use or implementation of the prototype could rely on them.

Coronary heart or artery disease (CHD) is a narrowing of the small blood vessels that supply blood and oxygen to the heart [171]. CHD is usually developed when a combination of fatty materials, calcium, and scar tissue are accumulated in the arteries that supply the heart with blood. Through these coronary arteries the heart muscle gets the oxygen and other nutrients it needs to pump blood [89]. CHD, the most common heart disease, is one of the leading causes of mortality in America and Europe [176] and it kills more Australians than any other single disease [116].

As the sector and the main features to monitor in the ongoing assessment of coronary heart disease patients were identified, the level of importance for the sub-list obtained previously was then needed given customized alerts were required. Thus, each cardiologist was requested to classify the sub-list provided by adding to it a level of importance. The level of importance is based on the risk it will represent for a patient that is affected by it. Even though full agreement was not obtained given each patient might react differently to each symptom depending of external factors such as health history, environment, age, sex, fitness, and if the patient was a smoker or not, based just in the sub list provided a general agreement was obtained as a starting point for the prototype implementation.

The features considered relevant to monitor in patients with coronary heart disease classified by level of importance can be seen in Table 5.1.

It is considered relevant to emphasize that the prototype has been built to monitor heart disease patients that have already been diagnosed as coronary heart disease patients.

Coronary heart disease diagnosis usually includes the assessment of 4 conventional risk factors which include the assessment of cholesterol levels, blood pressure levels, and if the

Table 5.1: Symptoms

Level of Importance	**Symptom**
High	ST-T alteration
	Dyspnea
	Hypertension
	Discomfort, heaviness in the chest
	Chest Pain
	Neck venous return or engorgement
Medium	Cyanosis
	Systolic murmur
	Dizziness
	Diastolic murmur
	Blood pressure
Low	Headache
	Second heart sound
	Barrel chest
	Upper respiratory infection

individual is a smoker or a diabetic patient as these feature raise the chance depending on the age to develop coronary heart disease. Nevertheless, even though these are features relevant for the diagnosis of coronary heart disease they were not all considered relevant for the ongoing assessment of patients already diagnosed with CHD [89]. It is usually suggested that more than 50% of patients with CHD lack any of the conventional risk factors and therefore there are other features and conditions that play a significant role in the development and prognosis of this disease and might be particular to each individual diagnosed.

The sector, in this case the coronary heart disease sector, and the main features to monitor in the ongoing assessment of patients with CHD were identified. More precise information about a list of episodes (combination of symptoms) that would trigger alarms for a particular patient was then needed. Cardiologists were able to identify six main episodes to be used in the prototype. Although this list is obviously incomplete, it was considered adequate to demonstrate the validity of the approach. A list of the episodes considered for the prototype can be seen in Table 5.2.

It is important to highlight that the range in which a heart disease feature should move is well defined in the literature, i.e. normal blood pressure data ranges must be less than 120 for a systolic reading, and less than 80 for a diastolic reading. Nevertheless, each patient

might have conditions which do not necessarily fit in the base of knowledge of heart disease. As an example patient normal blood pressure reading (value) differs from one patient to another. Thus, source agent has to contrast for the pre-analysis the general heart disease base of knowledge as well as the patient specific data values for each critical feature relevant to monitor.

Table 5.2: Coronary disease episodes

Episode Name	**Symptoms combination**
Pathological cardiac remodelling	Cardiac enlargement + Dyspnea + Systolic murmur
Cardiac insufficiency	ST-T alteration + Chest pain
Malignant ventricular arrhythmia	Syncope + Palpitations
Myocardial infarction	ST-T alteration + Third heart sound
Ischemia of the papillary muscle	New systolic murmur + ST-T alteration
Right ventricular infarction	Cervical venous engorgement + ST-T alteration + Chest pain

Thus, sector, entities, and the main features to monitor were identified and thus the main characteristics of the domain base of knowledge to be used as part of the TTL prototype was obtained (See Fig.5.2).

5.1.2 Prototype main mechanisms

There are three main functions or mechanisms that were considered important to implement in the prototype at this stage given they represent the core functionalities within the framework proposed here. These can be generally described as follows:

- A function to get the knowledge from the sources
- A function to use the source knowledge to pre-analyse data
- A function to transfer data to the warehouse after the the pre-filter/analysis mechanisms have been activated.

The first two functions belong to the local decision unit and therefore were used to achieve local empowerment. The third function was used to achieve data transference (data push) to the warehouse given the need to demonstrate that the data being pre-filtered could be

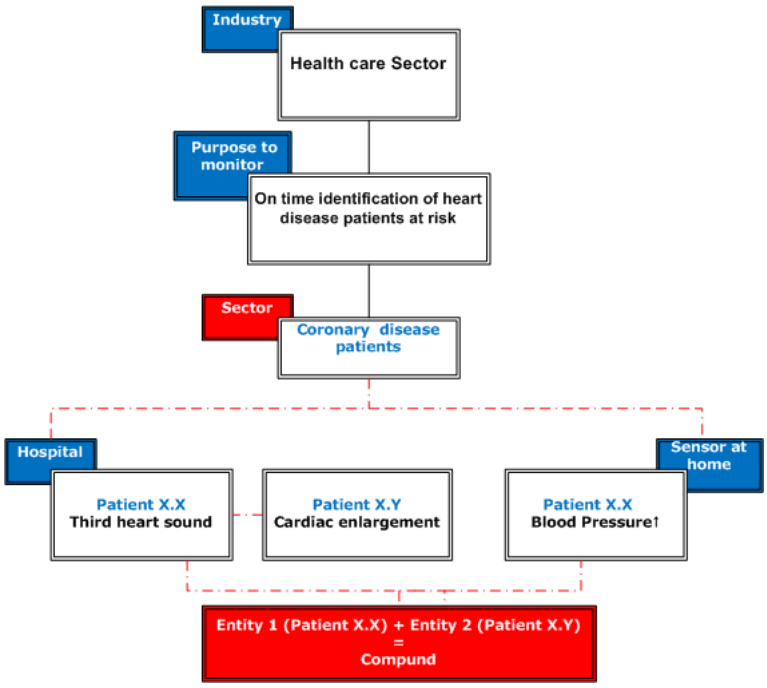

Figure 5.2: TTL Prototype general base of knowledge description

sent to the central repository for further analysis or for alerting of users. Even though there are more core functionalities that can be implemented, for example to deploy the feedback mechanisms and knowledge updates of the source agents, at this stage these will not be considered necessary to prove the concepts and answer the research questions of this research.

A description of how these 3 core functions were deployed is given in the following sections.

5.2 Implementation of a local decision making unit

Based on the base of knowledge given by experts in the field (as described in the previous section) and a set of features considered to be relevant for the ongoing assessment of patients with coronary heart disease, a simulated environment of patient scenarios, disease data and cases, was deployed.

Even though a long time was dedicated to attempting to obtain real patient data (health records), for security reasons and the lack of patient consent this option was not viable. Health care records are protected by law and the disclosure of such information requires

primarily patients consent. Thus to request individual health records in Australia, even for research purposes, it is necessary to undergo a series of time consuming activities such as meetings, ethics clearance, and filling a number of forms to request a permit to use and obtain anonymized patient data [66].

The main characteristic of the local decision unit is to be able to capture the changes made (update and/or insertions for example) of all the relevant data defined to be monitored at the source level. Once relevant data changes have been detected and captured a pre-analysis is required. The pre-analysis pre-filters the data that the warehouse will contain and according to pre-defined decision rules is able to recommend as soon as knowledge becomes available what might be the best decision to take according to the findings.

For the implementation of the local decision unit a PostgreSQL relational data base was built. PostgreSQL was chosen primarily because it is a non commercial application, multi-platform , and it does provide enough documentation and capabilities as a DBMS to build a prototype for research purposes. The PostgreSQL version 9.0.7 was used to store simulated patient's data, episodes and cases from the last 10 years for 20 coronary heart disease patients. The simulation of data was very simplistic, 20 coronary heart disease patients were created with a registry of their symptoms and disease episodes (combinations of symptoms as described in the previous section) in determined dates. The outcomes and management of the symptoms for each particular patient (i.e treatment, surgery) were also considered.

The simulation was validated by presenting samples of the data base content to the decision making users panel, who agreed with the way in which patients health care records were stored. It is assumed that for research purposes the simulation of health care records for 20 heart disease patients used is complete, accurate and represent patient states for the years 2001 to 2011.

5.2.1 Data types and data base schemas

Individual health records can be stored in different ways. Narrative as a summary of physical examinations or medical reports, numerical measurements such as laboratory tests and/or sensor device readings and recorded signals like in the case of ECG tests, and X-ray pictures,

ultrasounds and/or scanner tests.

Data types for the symptoms described as relevant to monitor in coronary heart disease patients are given in the following table (Table 5.3):

Table 5.3: Symptoms data types

Symptom	**Data type(s)**
ST-T alteration	True/False */Boolean/* + written report */Character data/* + waves and interval recorder signals */Image file, Character data/*
Dyspnea	True/False *Boolean*
Hypertension	True/False *Boolean*
Discomfort, heaviness in the chest	True/False *Boolean*
Chest Pain	True/False *Boolean*
Neck venous return or engorgement	True/False *Boolean*, Laboratory report *Character data*
Cyanosis	True/False *Boolean*
Systolic murmur	True/False *Boolean*, ECG report *Character data* + waves and interval recorder signals *Image file, Character data*
Dizziness	True/False *Boolean*
Diastolic murmur	True/False *Boolean*, ECG report *Character data* + waves and interval recorder signals *Image file, Character data*
Blood pressure	Numerical readings *Mostly Integer*
Headache	True/False *Boolean*
Second heart sound	True/False *Boolean*
Barrel chest	True/False *Boolean*
Upper respiratory infection	True/False *Boolean*, Laboratory report *Image file, Character data*

There are various levels of severity among the different symptoms and features a coronary heart disease patient could present. Depending on that classification alerts and treatment are defined. To simplify the prototype implementation the classification of only blood pressure stages was considered [67] (See Table 5.4).

Table 5.4: Blood pressure stages classification

Category	**Systolic**		**Diastolic**
	(top number)		(bottom number)
Normal	Less than 120	and	Less than 80
Pre hypertension	120-139	or	80-89
High blood pressure			
Stage 1	140-159	or	90-99
Stage 2	160 or higher	or	100 or higher

Depending on the classification of symptoms severity and the number of episodes a pa-

tient has presented along his/her disease prognosis and features such as age and gender a classification of patient risks is also needed [171]. From a medical perspective any patient that has presented an episode is classified as a high risk patient [176] and must be treated as soon as possible. Therefore, there are a number factors to consider while classifying patients at risk.

The risk classification for coronary heart disease patients used in this research is based on those features that get the patient to a point in which his/her life is in danger and a prompt response from the health care staff is needed. Therefore high risk patients are likely to present life threatening conditions at some stage during the prognosis of the coronary heart disease.

Some features increase the risk factor such as age and gender. Usually men have a higher risk of getting heart disease and present more heart episodes while already diagnosed with coronary heart disease than women who are still getting their menstrual period. After menopause, women's risk gets closer to men's risk. Smokers, diabetes and chronic kidney diseases, high blood pressure and being overweight increase the risk of heart failure in patients too [89].

For the prototype implementation and using the research presented in [176] [89] and [178] as a basis, a risk factor for each patient was calculated. This was based on the following rules (See Table 5.5):

Table 5.5: Cholesterol

mg/dl	**mmo/L**
<160	<4.14
160-199	4.15-5.17
200-239	5.18-6.21
240-279	6.22-7.24
≥ 280	≥ 7.25

Table 5.6: Blood Pressure

Systolic	**Diastolic**				
mm Hg	<80	80-84	85-89	90-99	≥ 100
<120					
120-129					
130-139					
140-159					
≥ 160					

Table 5.7: Diabetes

Diabetes
yes
no

Table 5.8: Smoker

Smoker
yes
no

Table 5.9: Blood pressure stages classification

Symptom Importance level	High	Medium	Low
High			
Medium			
Low			

Table 5.10: Patient risk factors legend

Legend
High risk patient
Moderate risk patient
Low risk patient

The risk factor, as displayed in the previous tables, was decided to be low, moderate and hight. Any coronary heart disease patient has then a risk factor associated given the conditions, diseases and symptoms the patient has presented since the time they were diagnosed. The risk factor helps with the alerting mechanism as it gives an insight of how quickly a medical decision needs to be taken.

As mentioned previously, heart disease normal parameters are well defined in the literature, but each patient might have conditions which do not necessarily fit in the base of knowledge of heart disease. As an example patient normal blood pressure, the actual value and ranges, differ from one patient to another. Thus, source agent has to contrast for the pre-analysis the general heart disease base of knowledge as well as the patient specific data values for each critical feature relevant to monitor.

An example of the data base schema for the GP data base simulated and the data warehouse design used for the prototype are provided in Fig. 5.3. and 5.4.

A general description of the data base design used can be described as: A *Patient* visits a *Doctor* on a determined date, *Patient* presents a number of *Symptoms* or *Episode* which are recorded as part of each visit. A *Patient* might present *symptoms* which are recorded but not necessarily attached to a doctor visit. A Patient has *Sensor* data and a number of *Symptoms* coupled together to form a disease *Episode*.

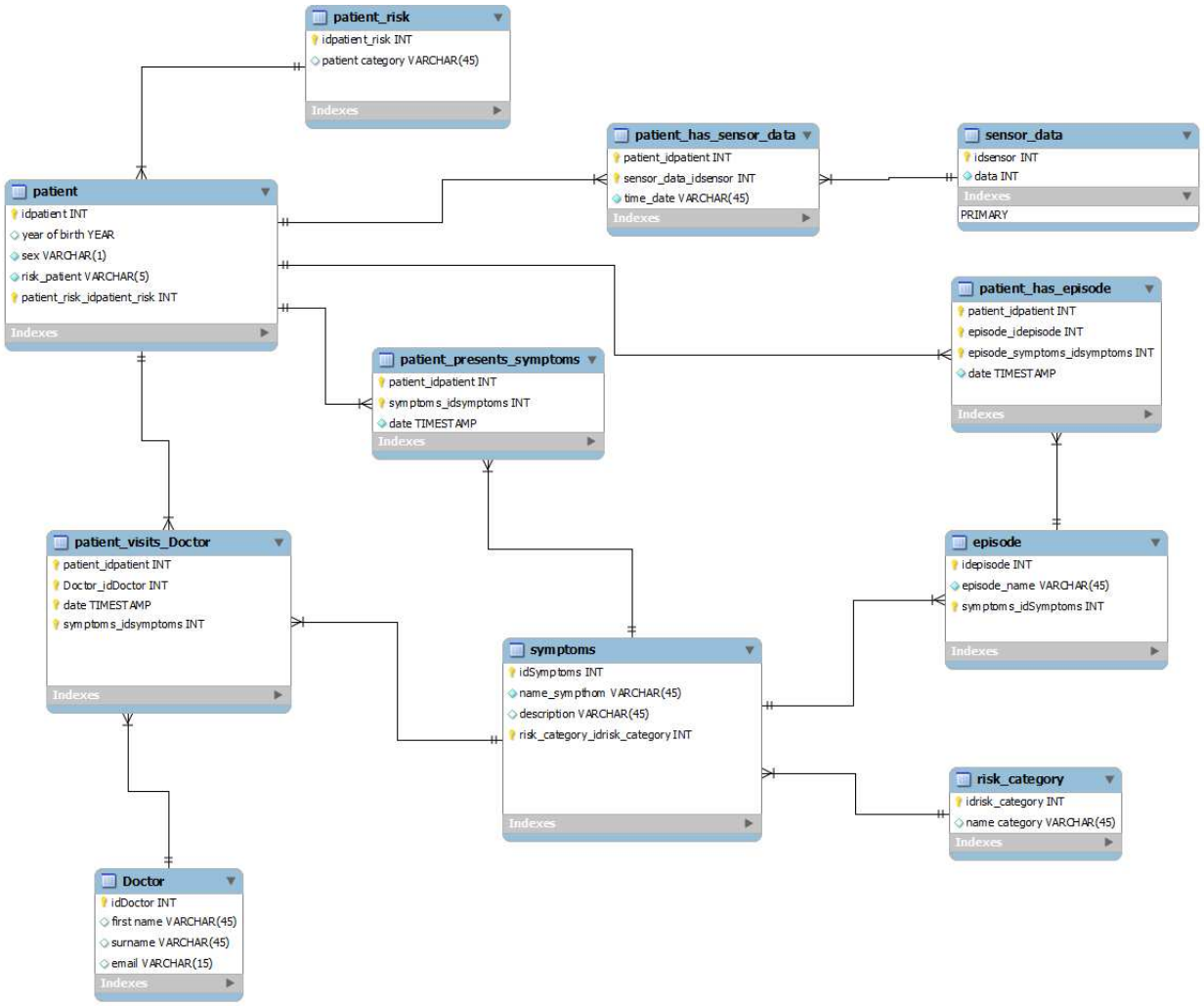

Figure 5.3: GP data base

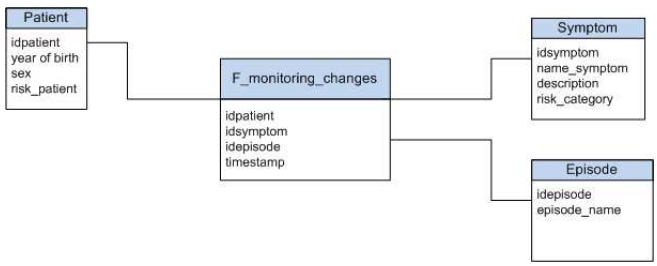

Figure 5.4: Data warehouse design- star schema

The data warehouse design chosen was the star schema. The star schema can be described as the way it models the end user's view by providing a natural mapping between the sector entities being monitored by source agent and the schema design. Usually, changing dimensions in a star schema might be easier than in a snowflake schema, star schemas enable high speed access for large volumes of data and are comprehensively supported by a vast number of business intelligence and reporting tools [74][57].

Once the data types, database and data warehouse schemas were designed and implemented, the deployment of the mechanism to empower the data sources was needed. A discussion of how this mechanism was implemented is provided in the following section.

5.2.2 Empowering the sources of information

There are many ways to extract knowledge from data sources. Most approaches focus on the use of data mining tools/algorithms for knowledge discovery in data bases such as research presented in [74][148][85] and [182]. Given the time frame this research had and because the idea of developing a prototype was to prove the concept for the framework proposed, a different and a more simplistic approach was used.

Once patient data was simulated, and the coronary heart disease base of knowledge to be used as part of the prototype was obtained, a function was programmed in a query to extract the knowledge about each patient from the sources, in this case to obtain the patient's data accepted ranges.

To extract patients knowledge the use of a couple of aggregate functions was needed. Most DBMS support aggregate functions. In the case of PostgreSQL the extraction of the maximum value and minimum value *max* (maximum), *min* (minimum) per row per patient was needed to get the maximum and minimum values recorded for blood pressure (See Query 5.1). Those minimum and maximum values will represent the accepted ranges for blood disease for each patient.

Then another query was made to return the patients identity for each patient that has presented an episode and/or has diabetes, smokes and has a high cholesterol level. Thus, the necessary information was obtained to start with the base of knowledge of source agent for

each patient to be monitored.

There were three main steps to empower each data source to monitor patients data (See Figure 5.6):

- Placing a trigger to look after any data changes for a particular patient
- Have source agent ready to obtain the trigger outcome
- Enable source agent to mine the changes and use the base of knowledge to perform an action, in this case send an alert and/or update the data warehouse.

Figure 5.5: Functions used to empower the data sources

5.2.2.1 Trigger function

The main idea to implement the monitoring mechanism was having source agent subscribed to patient data base changes. PostgreSQL provides a LISTEN / NOTIFY functionality which notifies data base events with an optional *payload* string to each client application that has previously executed LISTEN channel for the specified channel name in the current database. The function takes the channel name as the first argument and the payload as the second. The function *pg_notify*(text, text) makes this simple by providing a publish and subscribe mechanism within PostgreSQL.

There were two options to be notified of patient data changes. One option was creating a trigger with the *pg_notify* function and have the source agent in a Java program as a listener

so it could connect and poll PostgreSQL patient data changes every millisecond to see if there were new notifications. Something to have in mind is when using the JDBC driver is that there is no way to receive notifications asynchronously, therefore a poll mechanism must be used.

Even though the polling notification is light weight and can be programmed to query data frequently (i.e a poll (400) will ask 400 times in a second), it can be seen that the source agent will be pulling frequently to get data changes and a pushing rather than a pulling action was required.

The second option was creating a trigger with the *pg_notify* function (See Algorithms 4 and 8) and have the source agent in a C program listening to patient data changes. This option was selected given the option to communicate between processes (socket programming) that C provides. Thus, a pushing mechanism from the data sources was obtained and therefore the source was empowered to send patient data changes to a source agent.

The function and the trigger mechanism are given as follows:

Algorithm 4 Trigger function

```
1: CREATE OR REPLACE FUNCTION patientchanges()
2: RETURNS trigger AS $$
3: DECLARE
4: BEGIN
5:     PERFORM pg_notify(CAST('patient_has_symptoms' AS text),CAST(NEW.patientid
   AS text));
6:     RETURN NEWDATA;
7: END;
8: $$ LANGUAGE plpgsql;
```

Algorithm 5 The Trigger

```
1: CREATE TRIGGER patientnotify AFTER INSERT OR UPDATE ON pa-
   tient_has_symtoms
2: FOR EACH ROW EXECUTE PROCEDURE patientchanges();
```

The function used in the trigger is called ***patientchanges()***. It uses pg_notify to capture patient data changes, in this case inserts and/or updates in the table ***patient_has_symptoms***. *NEWDATA* returns the changes along with the name of the attribute modified. CAST allows typecasting to the data captured as text.

5.2.2.2 Pre-filtering data - C program

Source agent keeps waiting to listen from patient_notify data capture, and as soon as any data change is received it checks whether the change is relevant for the patient to monitor. If the data change is relevant a further action is taken, in this case alerting by sending a message to the health care staff responsible for the patient (decision maker) and/or inserting the data in the data warehouse. An overview of the main functions used to achieve data pre-filtering/alerting and transference that source agent uses are shown in Figure 5.6.

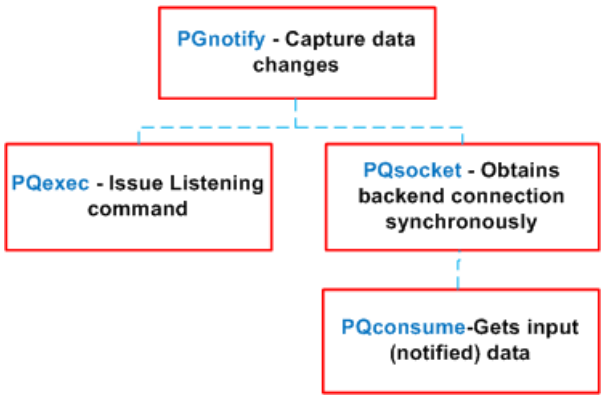

Figure 5.6: Functions used to empower the data sources

The four main functions showed in Figure 5.6 were the ones used to empower the sources of information and to provide source agent with the right information to perform the data pre-filtering/alerting. These functions were programmed in C and obtained from the Libpq library [131].

Libpq is the C application interface for programmers built in PostgreSQL. It contains a set of library functions that let client programs pass queries, and receive the results of these queries, over to the PostgreSQL backend server [130]. There are several PostgreSQL functions that can be used in a C environment.

A view of the main functions used for the empowering mechanism is given in Algorithms 6, 7 and 8.

Algorithm 6 shows the way the LISTEN command was issued. PQexec allows submission of a synchronous query to patient_has_symptoms. A synchronous query maintains the process

Algorithm 6 Source agent - Enabling notifications (C code)

```
1: ...
2: /* LISTEN command ready to allow notifications */
3: res = PQexec(conn, "LISTEN patient_has_symptoms");
4: if (PQresultStatus(res) = PGRES_COMMAND_OK) then
5:     fprintf(stderr, "The LISTEN command implementation has failed:
6:     %s", PQerrorMessage(conn));
7:     PQclear(res);
8:     exit_nicely(conn);
9: end if
10: ...
```

of the application controlled for the duration of the query. PqresultStats provides the result status of that query, while PQclear clears the results associated with the query once it has failed.

Algorithm 7 Source agent - Waiting for input (C code)

```
1: ...
2: int sock;
3: fd_set input_mask;
4:
5: sock = PQsocket(conn);
6:
7: if (sock < 0) then
8:     break;
9:     FD_ZERO(&input_mask);
10:    FD_SET(sock, &input_mask);
11: end if
12: if (select(sock + 1, &input_mask, NULL, NULL, NULL) < 0) then
13:    fprintf(stderr, "select() connection has failed, or no connection was established:
14:    %s
15: n", strerror(errno));
16:    exit_nicely(conn);
17: end if
18: ...
```

PQsocket obtains the file descriptor number of the back end connection socket of the source of information where the information must be captured. A number ≥ 0 is accepted and correct while a -1 implies that no connection was established/open.

In Algorithm 8, all the readable data on the file descriptor identified by PQsocket will be retrieved and ready to be consumed by source agent.

PGnotify passes its contents to the notify variable. It provides us with the channel name,

Algorithm 8 Source agent - Retrieving input (C code)

```
1: ...
2: PQconsumeInput(conn);
3: while ( do(notify = PQnotifies(conn)) = NULL)
4:     fprintf(stderr, "Notification of '%s' received from backend PID is %d and the PAY-
   LOAD is %s ,
5:     data captured is ready to be analysed
6: n",
7:     notify− >relname, notify− >be_pid, notify->extra);
8:     PQfreemem(notify);
9:     nnotifies++;
10: end while
11: ...
```

the ID of the notifying server, and the payload string. For the prototype the channel has the same name of the table to monitor in the database, in this case patient_has_symptoms.

Source agent gets the data captured (notification and data changed) and it proceeds with the data pre filtering in the following order, Algorithm 9.

Algorithm 9 Source agent - Checking input

```
1: Check which type of data has changed
2: if Data changed is relevant then
3:     Check the content of the data
4:     if data changed is not under normal ranges then
5:         notify health care provider and insert data in the data warehouse
6:     else
7:         free data captured and clean variables
8:     end if
9: end if
```

As an example in C code the pre filtering mechanism for a change in systolic blood pressure will be as follows (Algorithm 10):

Once source agent has been given the data changes captured, in this case a change of systolic blood pressure for patient id.435, it retrieves the system time stamp by using now(), one of PostgreSQL date functions. This is done to identify when the data change (event) has been detected.

Then it compares the data changed with the current systolic blood pressure accepted ranges for patient id.435 and proceeds with the predefined rules to monitor the blood pressure. An email, using the predefined function pgmail, is issued once the content of the data changed

Algorithm 10 Source agent - Checking input-Systolic Pressure change

```
1: ...
2: SELECT now();
3: strcpy(time,now());
4: int sys_435[2];
5: sys_435[0]=110;
6: sys_435[1]=129;
7: new=atol(notify->extra)
8: if new>=129 && new <= 139 then
9:     select pgmail('@monitoring','@patientid435 ','Device monitoring alert','We recommend you to have some rest and take your blood pressure again in the next 15 minutes.');
10: else if new>=140 then
11:     select pgmail('@monitoring','@healthcare ','Patient monitoring alert','Patient ID.435 Systolic Blood pressure is considerably high.');
12:     EXEC SQL INSERT INTO patient_multidimentional VALUES (:id.435 ,:new ,:time);
13: else if new<129 then
14:     select pgmail('@monitoring','@dba ','Data changes','some relevant data changes but tolerable?');
15: end if
```

has reached the high risk classification for the patient. This is done along with the transfer of the data change, patient id and time stamp to the data warehouse.

When the data changed is within the low/medium risk factor an email is issued to verify first device errors and/or false positives given that high systolic blood pressure can be altered because of exercise, medication and/or stress.

5.3 Implementation discussion

There were different ways to implement the TTL architecture. One of the first approaches in mind was to implement the local decision unit by using Oracle 9i and the change data capture (CDC) mechanism provided by it. Change data capture identifies and captures data that has been removed, updated or added to any Oracle relational table. By using CDC there were two possible methods; synchronous CDC, and asynchronous CDC.

For synchronous CDC triggers are used to capture change data and changes are published in a table called changed table. Asynchronous CDC works with the use of redo log files instead in which change sets are populated as soon as it commits for new transactions are detected. Both methods present the problem of impacting the source database transactions,

need the interaction of a DB administrator and both methods require a staging database that must reside at the source data base and must use the same operating system, Oracle release and hardware which limits the application of the TTL framework to just the Oracle domain. Thus, these options were not viable to be implemented as one of the core characteristics of the TTL architecture is to detect abnormalities in an unsupervised (no human interaction) way by empowering the sources to detect those changes and act upon them

After the Oracle attempt to build the prototype, the idea of working with open source technologies was then proposed given the liberty to modify and adapt the code and functionalities of data bases such as MySQL and PostgreSQL in order to fulfil the TTL main architecture requirements of data pre-filtering and transference through the empowerment of the data sources.

Both MySQL and PostgreSQL are widely recognized data base management systems that strongly compete with proprietary database software such as Oracle and provide enough documentation and reliability to implement the TTL architecture.

PostgreSQL was chosen over MySQL for the following characteristics [11][12]:

- the ability to support complex queries, asynchronous commits and asynchronous replication.
- the extensive range of aggregated functions and predefined commands to extract, mine and manage the data stored
- the integration option with external tools or applications for OLAP and data mining.
- the ability to run stored procedures by providing a native programming interface for programs such as Java, Perl, Python, Ruby, C/C++, and its own PL/pgSQL.

Due to the fact that the main purpose of the prototype was to empower the sources of information to build the decision making unit, the use of a specific software agent environment/platform was not needed. By definition a software agent is a software program with the ability to react to the environment, has autonomy, and has goal-orientation [55]. In these terms the program built in C code has the goal to empower the sources by reacting to data

bases changes (environment) and it is completely autonomous to capture, pre-analyse and transfer the data changes to any data warehouse that provides a C interface such as MySQL, PostgreSQL and Microsoft SQL server.

Figure 5.8 shows a full view of the prototype implementation.

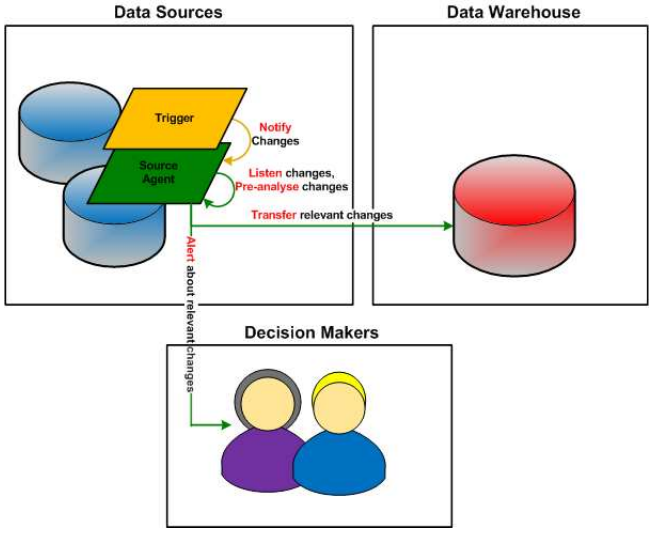

Figure 5.7: Functions used to empower the data sources

The prototype captures data changes at the source level by using a trigger that publishes data changes in a synchronous way to source agent. Source agent receives the changes, pre-analyses them and acts upon any abnormality presented by notifying the users and/or transferring this data to the data warehouse.

Chapter 6

Concept Validation - Research Findings

This chapter validates that the prototype model achieves the outcomes identified as contributions for the TTL architecture proposed. The overall aim of this research is to describe *a smart and rapid response mechanism to analyse, process and monitor data to detect data abnormalities and/or risk scenarios in real time using data warehouses.* Thus, to prove that the local empowerment of data sources to pre-analyse and push data to feed data warehouses enables better response and alerting mechanisms to detect risk scenarios in real time, a prototype was created and evaluated in terms of efficiency and effectiveness. Efficiency is used as the way to do the task using less resources and effectiveness by means of doing the task in the right way.

The main outcomes for the tests performed to evaluate the prototype and the architecture itself are discussed and presented in the following sections.

6.1 Decision making effectiveness

As was presented in the first chapters of this dissertation, one of the reasons for which organizations pursue the use of a business activity monitoring mechanism is to know *on time* what to do when opportunities, risks, advantages and any other relevant states that will af-

fect the organization in some way occur. Data is then continuously monitored as a decision support tool. Therefore, the first prototype assessment focused on validating the efficiency of the TTL framework prototype to support decision making in health care. To do that a qualitative approach was followed.

It was decided to include in this evaluation four main dimensions to validate the effectiveness of a patient monitoring mechanism in health care. These areas were; improving quality of care, improving quality of life for patients, saving time of healthcare professionals, and modernizing health care delivery. These areas were taken from Arah et al. [7] which research gives an overview of the main areas to consider to validate the effectiveness, quality and some other ways of improving performance of health systems in e-health.

The dimensions were considered to be assessed in the following way:

- Improving quality of care (IQC) was defined as to whether the TTL prototype provides easier, faster access and the right patient data.
- Improving quality of patient life (IQP) was defined as how well the TTL prototype monitors patient's heart disease conditions by providing alerts as soon as patient data abnormalities are found, and consolidates individual health records to support the delivery of tailored medical support.
- Saving time (ST) was defined as to whether the TTL prototype provides a supporting decision making mechanism to health care staff.
- Modernizing health care delivery (MHC) was defined as to whether TTL brings a degree of sophistication to the health care systems by allowing a faster flow of information for continuous care.

A walkthrough can be used as a way to review the findings of an investigation and the models that have been built based on those findings [149]. Thus, a structured walkthrough was used as a way to identify improvements in the monitoring process and to evaluate at the same time if the level of compliance of the prototype with the requirements defined was achieved. A panel of users, 8 cardiologists, was formed. The main idea was to determine the

level of satisfaction, in their opinion as to how efficiently the TTL prototype performed, in each of the four dimensions defined previously. To do that four scenarios were tested.

- The first scenario was to alert to an unusual recording of a high level of importance symptom, in this case source agent should react to an unusual recording of blood pressure (rise of blood pressure).

- The second scenario was alerting to an unusual recording of a medium level of importance symptom, in this case source agent should react to a patient that is informing of dizziness.

- The third scenario was detecting a data error, in this case source agent should check that the content of the data received did not correspond with the data type defined (different data structure for the systolic blood pressure).

- The last scenario was detecting a patient episode, in this case source agent should recognize that the number of symptoms recorded meant that the patient was presenting an episode.

In the walkthrough a developer and the eight specialists (reviewers) were present. An overview of what was expected from the walkthrough was given along with an explanation of each dimension to validate. Each scenario was simulated by a developer who provided each user with the outcome of each test performed. The main results of the walkthrough in terms of user acceptance for each dimension are displayed in the following table. Table 6.1.

Table 6.1: Efficiency test results

Scenario	**Detection**	**IQC(%)**	**IQP(%)**	**ST(%)**	**MHC(%)**
Blood pressure	yes	50	62.5	37.5	75
Dizziness	yes	50	100	37.5	75
Data error	yes	87.5	100	100	100
Blood pressure and Dizziness	yes	100	100	100	100

The results showed in Table 6.1 showed the percentage, on average, of the acceptance rate of the expert panel for the TTL prototype in all the test cases considered.

The reviewers discussed the outcome of each test altogether to have a view based on their peers and own experience to decide whether the outcome and detection of a high blood pressure rise, dizziness, the presence of an episode, and an error detection could be accepted or not. Some reviewers did not accept the outcome while some others did. The table shows the level of acceptance for the total of reviewers for each test and dimension of effectiveness evaluated.

As an example a level of 50% of acceptance for test number 1, high blood pressure, in the first dimension means than half of the reviewers, just 4 specialists, agree that the detection and alerting mechanism used (outcome for the test) for monitoring a patient with cardiac heart disease represent an improvement in the quality of care (IQC).

As can be seen TTL performs well in terms of effectiveness of decision making with a 100% of acceptance rate to identify errors and patient episodes. The outcomes for the test performed in the four areas of effectiveness were considered unanimously *accepted as it is* by the expert panel. This means that the TTL prototype provides with an efficient tool to detect data content abnormalities in patient monitoring and in detecting patient events given the knowledge source agent has about the patient, and the general area of knowledge in this case coronary heart disease.

However, according to the expert panel the identification of individual abnormal behaviours for test 1 and 2 (detection of a change of high and medium importance) needed more refinement, as the action to take in a change of symptoms in this case depends on an individuals environment conditions given the outcome could represent a false positive alert. Hence, a patient's blood pressure could rise because of movement, stress and other factors that do not really represent an emergency and therefore more information for this unusual behaviour must be gathered. The panel could not completely agree on the efficiency of the alert mechanism tool to detect these types of scenarios and thus only the detection of it was accepted as efficient.

In health care environments the inclusion of environment conditions that could affect and/or gather the detection of an unusual behaviour at the time of monitoring might need to be included in the future as part of the whole framework in order to not give false positive

alerts. As an example a patient might present high blood pressure while under stress or walking quickly which does not necessarily represent a risk condition.

6.2 Computational efficiency

In a system designed to actively monitor data, resource consumption and response time are usually the metrics of interest to evaluate a system.

To measure computational efficiency an experiment was set up by using a dedicated machine in which a PostgreSQL data base with 20 heart disease patients, with records from the years 2000 to 2010 was simulated. Data in the database includes GP patient records (visits, symptoms, outcomes/events, and medication), pathology results (test, ECG results), hospital patient records and sensor monitoring data results. That gave us an environment of around 6000 records for the total of patients.

We used Linux Ksar utility to monitor the system states and resource usage during the tests in order to compare traditional data extraction vs intelligent transference and pre-analysis. The results of these tests can be seen in Table 6.2

Traditional extraction was considered as a simple planned and batched programmed query that extracts a patient's data to be moved to a warehouse in a certain period of time. Intelligent transference is the new approach that we propose. A simple scenario refers to a scenario in which data has changed at the source level but is not relevant to monitor. A complex scenario refers to a new data entry at the source level that is relevant to monitor and an action must be taken.

Table 6.2: Cpu usage

	Scenario	Duration in sec	Average usage of CPU in %		
			User	System	Idle
Traditional extraction	Simple	1.002	50.29	19.27	30.44
	Complex	1.005	51.27	18.29	24.52
Intelligence transference	Simple	0.637	30.27	21.15	48.58
	Complex	1.297	44.17	19.31	36.52

Traditional extraction was considered as a simple planned and batched programmed query

that extracts patient's data to be moved to a warehouse in a certain period of time. Intelligent transference is the new approach that we propose. A simple scenario refers to a scenario in which data has changed at the source level but is not relevant to monitor. A complex scenario refers to a new data entry at the source level that is relevant to monitor and an action must be taken.

As can be seen in the table the intelligent transference, (agent pre-analysis) consumes less or almost the same amount of CPU than the traditional data extraction mechanism programmed. Therefore, enabling pre-processing and filtering at the source levels does not stress the sources of information. Source agent seems to run at a low priority (given the % CPU idle used) so does not impact programs that run at normal priority in the server.

In the complex scenario SA performed pre-analysis and a transference and it did not show mayor differences in relation to traditional transference.

These tests were planned to move a number of KB only (200 records) which is the best scenario and it does not represent a Very Large Data Base (VLDB) environment. Nevertheless, on a VLDB environment traditional data extraction will use more machine resources and it will be more time consuming [180] [107] while in our case the database size is not relevant as SA will keep almost the same average consumption as it will only analyse relevant data (a small data load) and not all the data that has changed in the source.

In terms of process duration traditional data extraction is affected by the number of rows/bytes to extract while intelligent data transference is affected by the amount of data to analyse. Because in the tests a small data load was simulated, it was not really possible to compare process duration with the two scenarios proposed. Therefore, although these data is included in the table they were not considered enough to be discussed in any depth here and further tests with a large dataset need to be done in the future.

6.2.1 Algebraic validation of efficiency

In traditional data warehouse environments every time data needs to be refreshed a query is sent to each source of information that takes part of the data warehouse architecture, then information is extracted and loaded in the data warehouse. A optimal loading process

depends on the organization requirements and how the data warehouse design was planned so it varies widely. Usually refreshments are planned on a daily basis, data is extracted and placed in the data warehouse by overwriting any data from the previous load and replacing it with fresh data. This is commonly known as a full load.

More complex systems currently pursue incremental loads by checking the date from the last load that was performed (timestamps), log files or even by using a change data capture mechanism where generally any data that has changed at the source level is retrieved and published in a new table. The data warehouse then queries that table to extract the data to be loaded in the central repository. Thus new data only is loaded in the data warehouse.

Thus, by selecting the use of an incremental load example and comparing it with the TTL architecture the amount of data to be loaded to the data warehouse, *DL*, will be as follows:

- ***Incremental data load***

  N: represents the number of sources to query

  n: represent all the new data (data that have changed since the previous load)

  t: is the time where data is queried at the source level, pulled, from the warehouse in order to refresh the data it contains

$$DL_t = \sum_{i=1}^{N} n_i \tag{6.1}$$

- ***TTL - Relevant data transference***

  K : represents the number of sources that capture changes in new data, $K \leq N$

  u : represents unusual new data, $u \leq n$, the data that has changed in a source that is not within usual ranges

  t': represents the time in which the data that has changed at the source level is pushed to source agent to be pre-analysed

$$DL_{t'} = \sum_{i=1}^{K} u_i \tag{6.2}$$

As can be seen, in the worst case scenario where all sources will capture new changes, and all of those changes are relevant (all the changes captured represent an unusual data

range), by looking at the equations the *TTL will pull the same amount of data changes that the incremental load will push to the warehouse.*

However, in most cases TTL will only pull the new data that has changed and has an unusual content. Therefore even though data has changed in all the sources it might be possible that only a portion of it, represented by u, will be transferred to the warehouse.

Another important characteristic of the TTL architecture to highlight here is that t, the data load time for an incremental loading is programmed. Thus, several or non data changes might occur at the source level. If several changes have occurred and those data changes show an unusual data behaviour, or unusual data content, they will not be picked up nor reported to the decision maker until the data analysis is performed which usually takes place after the data loading in the warehouse has finished.

Just taking as a example any business scenario, suppose that 5 sources of information feed a data warehouse. In source 1 after previous loading cycle 1000 kilobytes of new data has been stored, in source 2 - 3000 kilobytes of new data needs to be moved, in source 3 - 1 terabyte, source 4 - 500 gigabytes , and source 5 has remained the same (no changes).

$$DL_t = 1{,}000+3{,}000+1{,}073{,}741{,}824+ 500(1{,}048{,}576)$$

$$DL_t = 1524 \text{ GB approximately}$$

In an incremental load for this data refreshment scenario 1524.00 GB will be pulled to the data warehouse. In an intelligent transference instead, TTL will perform the data refreshment in a different way as even though maybe 1000 KB of new data records will be stored in source 1 from the previous refreshment, only 10% of them might represent an unusual reading and therefore an action will be needed. Thus, just a portion of the new data changed in source 1 will need to be transferred to the data warehouse.

If in average for the 5 sources only 35% of the new data that has changed is relevant given its unusual content, outside accepted ranges. The amount of data transferred to the warehouse will be:

$$DL_{t'} = 1{,}598{,}033{,}824 \text{ kilobytes } * (0{,}35)$$

$DL_{t'}$ = 533.40 GB approximately (Approximately 65% less than in an incremental load)

Hence $DL_{t'}$ is considerable less than DL_t, for the example provided 65% of the data changed will be only loaded in the warehouse. It is possible to say that $DL_{t'}$, TTL pre-filtering, is more efficient to feed a data warehouse than using ETL. The TTL will transport, transform and load less data into the data warehouse, i.e only what is needed to monitor.

Now evaluating in terms of processing the two previous scenarios, incremental loading and relevant transference, the equation that represents the analysis of information in terms of workload *(W)* can be described as follows:

- **Incremental data loading**: In an incremental data loading the data warehouse scans each source of information *(S)* and then retrieves all the data that has changed *(n)* to a staging area where data is transformed to then be loaded in the data warehouse. The staging area is usually located where the data warehouse resides. Therefore, the amount of work performed until reporting to decisions makers is being processed in the "machine" where the data warehouse is. Thus, if p represents the number of processes involved to analyse data in the data warehouse, which includes the data extraction, transformation and loading, it can be seen that W will be represented by:
  S: numbers of sources to extract data
  n: data that has to be moved in each source

$$W = \sum_{i=1}^{S} (n_i) * p \tag{6.3}$$

- **TTL - Relevant data transference**: In TTL the workload will be distributed between the data sources and the data warehouse as follows:

  - **Each source workload *SWk* is represented by**
    S: Number of sources to capture data changes
    m: data that has changed, m≤n
    p: numbers of processes to analyse data at the source

$$SWk = (m) * p \tag{6.4}$$

– **The data warehouse workload *DWk* is represented by**

r: relevant data that has changed and transported to the warehouse, r≤m

p': numbers of processes to analyse data at the data warehouse

$$DWk = (r) * p' \tag{6.5}$$

Evaluating the worst case scenario, in which the whole amount of data that has changed at the sources are relevant and therefore loaded in the data warehouse, it can be seen that the workload of the data warehouse for TTL, DWk will always be $\leq$ than W, the workload of data warehouse in an incremental loading.

For TTL data will then be captured, pre-analysed, and reported if needed, at each source of information first and only when unusual changes have been detected and further analysis is required data will be transferred to the data warehouse. The evaluation and validation of this last equation (6.4 and 6.5) is considered as part of future research where a real application and identification of the processes involved in the whole architecture can be really identified from a machine point of view.

In the following section an overview of the main findings of the architecture validation is provided

6.3 A discussion of the findings

We argue that by monitoring key features in each patient, pre-analysing and then if needed transferring them to a central repository the proposed architecture is more effective than traditional ETL and data warehouse architectures.

The main differences between a traditional clinical data warehouse system and our approach can be seen in Table 6.3

By enabling the use of local knowledge and empowering the data sources (using a monitoring agent) to make use of this knowledge we have reduced the numbers of steps to analyse

Table 6.3: ETL vs TTL

Processes	**ETL**	**TTL**
Data Extraction	1st stage	Does not apply Data changes are captured in each source as they occurs
Data Transformation	2nd stage	1st in conjunction with the pre-analysis
Data loading	3rd stage	3rd stage and only if needed
Data analysis/reporting	After ETL has been completed	Analysis 1st stage (Source level and data warehouse level) Reporting any time if needed (Source level and data warehouse level)

data. Data analysis has been moved to an early stage starting once relevant data to monitor has been changed in a source. That pre-analysis is performed now in the local source and if there is enough knowledge to perform an action, an alert mechanism is activated to either alert the patient because maybe the data that has changed in the source is incomplete (i.e sensor device data) or to the health care staff because the data that has changed is relevant to monitor and implies that the patient might be at risk.

Data inconsistency is detected in an early stage because of the knowledge about data types and data structure that *source agent* has in each source of information. Therefore, *source agent* knows data structures and types of each source. Thus, data inconsistency is picked up in relevant data as soon as the agent is notified. It checks data structures and data content values before performing any analysis which helps us to inform and not include uncompleted data as part of the transference to the data warehouse.

Therefore, by monitoring only relevant data in a distributed environment and by having local knowledge to check normal patterns it is possible to identify changes in the disease prognosis at an early stage and send an alert as soon as possible to the health care staff once unusual patient data, which might be a patient risk scenario, is captured.

Chapter 7

Conclusions and Further Work

7.1 Summary

A new approach to design a real time data warehouse architecture to monitor a variety of data sources in which traditional extraction, transformation and loading techniques do not apply is proposed in this research. As the data warehouse is not programmed for querying, the transformation, filtering and data loading mechanisms are driven by the sources of information. The local sources of information are the ones responsible then for capturing, pre-analysing and transferring only relevant data to monitor to the data warehouse. Thus, a *pushing* mechanism rather than a *pulling* approach is achieved.

Most monitoring architectures that use a data warehouse analyse data only once all o it has been consolidated in the central repository. It is argued that there is often enough knowledge in operational systems to pre-analyse data at an early stage. Consequently, by making use of this knowledge sources of information are empowered to pre-filter and capture relevant data changes and to take a decision of what to do next, send information to the data warehouse and /or alerts to decision makers, as soon as abnormal/unusual content in data changes are encountered.

However in some real time data warehousing situations, it might be needed to have a big picture of the data changes captured at the sources to detect business opportunities or threats. In these scenarios only part of the work is done at a single source level, each source pre-analyse

the data changes first to then push the relevant data findings to the central repository. Thus, when the data of the multiple sources monitored comes together in the central repository a complete analysis is done and if needed alerts are also generated from there.

By using an agent infrastructure, agents learn and respond to data events to be able to react to abnormal behaviours as soon as data is captured at any source level. In this way, valuable time for decision making is saved. The decision making starts from the owner of information, data source, and only useful data (data that needs to be monitored) is sent to a consolidated repository.

To validate the TTL framework the implementation of the local decision unit was considered relevant to prove the main concepts of data pre-filtering and empowerment of data sources to actively monitor unusual patient data.

The research question, "Does the local empowerment of data sources to pre-analyse and push data to feed a data warehouse enable better response and alerting mechanisms to detect risk scenarios in real time?" has been addressed by designing a Transformation, Transference and Loading mechanism (TTL) that was validated in two main aspects, effectiveness and efficiency.

In terms of Efficiency:

- Response time was improved because:
  - The number of steps to analyse data was reduced by empowering the sources of information to pre-analyse data as soon it has been captured at any source level. This reduces data analysis latency as decisions can be now taken at a very early stage. This is possible to achieve by using a multi agent software system.
  - Only the data that is necessary to analyse in real time is monitored. An agent resides in each data source and uses the knowledge of the sector and entity to monitor in terms of data content,and accepted data ranges to check abnormal events every time a data source update is detected.
  - Historical and sector knowledge was used as rules of monitoring. Through this way source agent knows when and how to react and who needs to know about any

unusual data behaviour.

- Error detection was improved because of:
  - The knowledge source agent has about data content. Source agent knows what the data should look like and while pre-analysing data changes, errors and inconsistency can be identified.

In terms of Effectiveness:

- The TTL architecture allows data fusion by using a mechanism to pre-analyse data that comes from different sources at the local level. Sources of information push data from any source to be consolidated in a central repository.
- The TTL architecture detects abnormal/unusual behaviours while monitoring business activities by using rules extracted from the sector and the data history of each entity to monitor. These rules are then used for a program, source agent, as knowledge to monitor. Thus, local and distributed decision making works together to better respond to organizational/operational data changes. Following the idea of smart sources a pre-analysis of meaningful data is undertaken at a source level to perform alerts and generate from the very beginning a chain reaction in the decision making process when risk scenarios are present.

The effectiveness of the framework proposed was assessed by testing it in the area of health informatics. A prototype solution to manage chronic coronary heart disease patients was proposed by designing an architecture that learns and reveals the disease activity patterns through day to day measurements, and the clinical history of an individual patient. The architecture also reacts in real time by sending alarms according to changes in those patterns, and can be adaptive to new system conditions and changes in health care requirements.

In a typical operational system there would be thousands of patients with diverse sources of information. In a consolidated system, data extraction will be performed for all these thousands of records before proceeding with data analysis. If there is not enough information to proceed with the analysis, it will only be detected once data has been collected and sent

to the central repository. There is a significant amount of data that needs to be extracted before it can be analysed.

Given the walkthrough used to validate the concept of the TTL prototype with specialists/users in the health informatics field, TTL was demonstrated to be a useful and effective mechanism to support decision making, by reducing data analysis and reporting/alerting latency, while monitoring unusual data that comes from a variety of data sources.

TTL was also demonstrated to be computationally efficient because of the amount of data to be transferred, transformed and loaded in the data warehouse is less or equal to any existing incremental approach used to update/refresh data in a central repository such as a warehouse. Moreover, TTL reacts as soon as data changes/updates have been detected at any source and therefore updates are not programmed and driven by the central repository as local knowledge and a pushing mechanism driven by the owner of information , the sources, is used instead.

7.2 Contributions

A smart and rapid response mechanism to analyse, process and monitor data to detect data abnormalities and/or risk scenarios in real time using data warehouses was created. By evaluating the minimum characteristics that an active monitoring system should have according to Hackathorn [64], it can be said that the TTL architecture is:

- *Intelligent*, to detect abnormalities and to react as soon as relevant changes have been captured at the data sources. Smart to learn from events whenever possible. This is carried out by the actions performed in the data pre-analysis, transference and consolidation tier.

- *Active*, to continuously be able to capture meaningful events, and to inform that they are present to decision makers as soon as possible. This is performed in the data pre-analysis tier.

- *Agile*, to perform mechanisms to react, alert, and to monitor in the shortest time possible. These are achieved in the data pre-analysis and alerts tier.

- *Adaptive*, in the sense of providing an infrastructure that can learn from abnormal behaviours and historical operational data in order to propose changes and updates in the knowledge and logical structure of the rules that drives the monitoring and loading of data in the data warehouse. The Data consolidation tier is responsible for that.

By monitoring key behaviours, evaluating and then responding to abnormalities only, the proposed architecture should be able to respond and/or alert to risk scenarios in a more effective way than traditional right time data warehousing strategies.

It is also necessary to highlight that traditional data warehouses architectures could also benefit from the use of smart data sources by adding an analysis component at each source level. Which ultimately will help decision makers to take sound decisions given the information provided by the whole system could be more accurate.

7.3 Limitations

A limitation for this research study might be not being able to obtain real anonymized data to be used to analyse the prototype results obtained. Patients data is very is classified as a very sensitive and personal information. Sadly even though a long time was used to get this type of data, this option was abandoned later on given the time spent "lost" in trying to obtain it.

Another limitation was not having the right technology, software package, to might be implement in a better way the prototype. Even driven data warehouses software packages for BI, event dependant rather than time dependant, are not widely available. Software packages such as Oracle Golden Gate are proprietary ,very expensive and black boxes to be used for research purposes. The package needed to be used but also personalised to perform the tasks proposed in this research. This is something that cannot be really done with proprietary software this is why a open source application was selected instead.

7.4 Future Work

In the short term future this research might concentrate on the implementation of a learning mechanism that allows the TTL architecture to be adaptive as part of future research work. The idea of automatically updating agents knowledge will enormously benefit the framework proposed to make the whole architecture more autonomous and adaptive. If the central repository is prepared and empowered to mine the knowledge and able to learn from the every day monitoring of data received, agents knowledge updates can be driven from there. Thus, some sort of intelligence can be placed at the data warehouse level to drive adaptation.

It is believed that the TTL facilitator agent, *knowledge agent*, might not only be able to integrate information in the central repository and to achieve intelligent data analysis, it will also be possible to empower it to know when and how new knowledge updates need to be performed by source agents at the local sources. Case based reasoning seems to be the right reasoning mechanism to achieve this sort of adaptation because of its ability to support the automatic acquisition of subjective knowledge by adding new cases or adapting old solutions to meet new requirements.

If we consider that the general knowledge of physicians is acquired from past experiences with cases they have treated themselves, colleagues, or publications, the thinking process of physicians based on general medical knowledge and previous experience with other patients defines a typical case.

Hence, by using case based reasoning source agents at the local level (sources) could have a set of individual patient cases for which it is necessary to provide an alert. On the other side a knowledge agent at the central repository could have a library of patient cases and events from which previous experience can be extracted and compared. Thus, by identifying common factors between a retrieved case and the target problem found at the time of monitoring a reasoning mechanism to perform accurate alerts and support for knowledge adaptation can be achieved.

Another interesting topic for future research is to find an effective way to deal with the propagation of relevant data changes. Sources of information may change while the monitoring/transference/loading is performed. This could cause refreshment anomalies and finish

with a central repository in an inconsistent state. It is believed that source agents can be prepared to deal with change data mismatches. However, the use of determined data propagation approaches, the way data is propagated from one or more sources, and/or queue data execution to deal with this type of scenario might be worth considering.

Another future research direction is to find a way to deal with false positives alerts and the need to know which external events can also trigger unusual data behaviours for an entity. Even though some external events cannot be controlled, it might be possible to gather more information, such as asking a patient for more information once new and relevant data changes have been captured at the sources level. This will help to deliver sound alerts to decision makers and users of the monitoring architecture.

Last but not least, particularly in the health informatics domain there is great discussion on the creation/implementation of standards to achieve interoperability and safety to manage individual health care records among a variety of health data sources. Thus, the use and study of standards to achieve safety and interoperability while implementing and empowering data sources to transfer individual health care records and events in distributed environments would be useful to study.

Bibliography

[1] J. Adzic, V. Fiore, and S. Spelta. Data warehouse population platform. *DBTel '01 Proceedings of the VLDB 2001 International Workshop on Databases in Telecommunications II*, pages 9–18, 2001.

[2] H.F. Ahmad. Multi-agent systems: overview of a new paradigm for distributed systems. *Proceedings of the 7th IEEE International Symposium on High assurance systems engineering*, pages 101 – 107, 2002.

[3] A. Albrecht and F. Naumann. Managing ETL processes. *VLDB '08 Proceedings of the 34th International Conference on Very large data bases*, pages 24–30, 2008.

[4] D. Allan. Oracle warehouse builder 11.2: Upgrade and migration paths. Technical report, ORACLE, 2010.

[5] T. Alsinet, R. Bjar, C. Fernandez, and F. Manya. A multi-agent system architecture for monitoring medical protocols. *ACM transactions*, pages 499–505, 2000.

[6] I. Ankorion. change data capture- efficient etl for real-time bi. *Information Management Magazine*, January, 2005.

[7] O. Arah, N. Klazinga, D. Delnoij, A. Asbroek, and T. Custers. Conceptual frameworks for health systems performance: a quest for effectiveness, quality, and improvement. *International Journal of quality in health care*, 15(5):377–398, 2003.

[8] A.M. Arigon and A.M. Tchounikine. A multiversion model for multimedia data warehouse. *Proceedings of the 6th international workshop on Multimedia data mining: mining integrated media and complex data in ACM database*, 2005.

[9] B. Azvine, Z. Cui, D. Nauck, and B. Majeed. Real time business intelligence for the adaptive enterprise. *Proceedings of the The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Service*, pages 29–40, 2006.

[10] J. Barateiro and H. Galhardas. A survey of data quality tool. *Datenbank-Spektrum*, 14:15–21, 2005.

[11] G. Bartolini. Data warehousing with postgresql. *European PostgreSQL Day*, 2009.

[12] J. Berkus. Really big elephants. data warehousing with postgresql. In *MySQL User Conference*, 2011.

[13] M. Birna van Riemsdijk, M. Dastani, and M. Winikoff. Goals in agent systems: a unifying framework. *International Conference on Autonomous Agents. Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 2:713–720, 2008.

[14] J. Bleiholder and F. Nauman. Conflict handling strategies in an integrated information system. *IIWeb workshop*, 2006.

[15] J. Bleiholder and Naumann. Data fusion. *ACM computing surveys*, 41:1–41, 2008.

[16] S. Bobek and I. Perko. Intelligent agent based business intelligence. *IV IInternational conference on multimedia and information and communication technologies in education*, pages 1040–1051, 2006.

[17] R. Bruckmer, B. List, and J. Schiefer. Striving towards near real-time data integration for data warehouses. *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 317–26, 2002.

[18] F. Burstein and S. Gregor. The systems development or engineering approach to research in information systems: An action research perspective. In B Hope and P. Yoong, editors, *10th Australasian Conference on Information Systems*, pages 122–134, 1999.

[19] S. Carlsson, S. Henningsson, S. Hrastinski, and C. Keller. Socio-technical is design science research: developing design theory for is integration management, information systems and e-business management. *Journal of Information Systems and e-Business Management*, 9:109–131, 2011.

[20] M. Castellanos, F. Casati, M. Shan, and U. Dayal. ibom: A platform for intelligent business operation management. *21st International Conference on Data Engineering*, 2005.

[21] M. Castellanos, A. Simitsis, K. Wilkinson, and U. Dayal. Automating the loading of business processes data warehouses. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 612–623, 2009.

[22] Y. Cervantes, L .and Lee, H. Yang, S. Ko, and Y. Lee. Agent-based intelligent decision support for the home healthcare environment. *Lecture Notes in Computer Science. Advances in Hybrid Information Technology*, 4413:414–424, 2007.

[23] S. Chandrasekaran, O. Cooper, A. Deshpande, M Franklin, J. Hellerstein, W Hong, S. Krishnamurthy, S. Madden, F. Reiss, and M. Shah. Telegraphcq: continuous dataflow processing. *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 668–668, 2003.

[24] S. Chandrasekaran and M. Franklin. Psoup: a system for streaming queries over streaming data. *The VLDB Journal The International Journal on Very Large Data Bases*, 12(2):140 – 156, 2003.

[25] E. Chavez and G. Finnie. Empowering data sources to manage clinical data. *23RD IEEE International Symposium on Computer-Based Medical Systems (CBMS 2010)*, 2010.

[26] E. Chavez and G. Finnie. *Data warehousing and knowledge discovery*, chapter TTL: A transference, transformation and loading approach for active monitoring, pages 124–135. Springer Berlin / Heidelberg, 2011.

[27] Li. Chen, W. Rahayu, and D. Taniar. Towards near real-time data warehousing. *24th IEEE International conference on advanced information networking and applications*, pages 1150–1157, 2010.

[28] T. Chieu and L. Zeng. Real time perfomance monitoring for an enterprice information management system. *IEEE international conference on e-business engineering*, pages 429–434, 2008.

[29] M. A. Chizner. The diagnosis of heart disease by clinical assessment alone. *Disease-a-Month*, 48:7–98, 2004.

[30] A. Chohra, N. Kanaoui, and K. Madani. Hybrid intelligent diagnosis systems. *Proceedings of the Computer Information Systems and Industrial Management Applications*, pages 115–120, 2007.

[31] P. Chountas and V. Kodogiannis. Development of a clinical data warehouse. *Proceedings of the IDEAS Workshop on Medical Information Systems: The Digital Hospital in IEEE database*, pages 8–14, 2004.

[32] S. R. Chowdhury, A. Birswas, and R. Chowdhury. Design, simulation and testing of an optimized fuzzy neuronal network for early criticaly diagnosis. In *IEEE first international conference on emerging trends in engineering and technology*, 2008.

[33] E. Coiera and R. Clarke. E-consent: The design and implementation of consumer consent mechanisms in an electronic enviroment. *Journal of the American medical informatics association*, pages 120–140, 2004.

[34] E. Comak, A. Arslan, and I. Turkoglu. A decision support system based on support vector machines for diagnosis of the heart valve diseases. *Computers in Biology and Medicine*, 37:21–27, 2007.

[35] Bobby D., LEE Jae-Wan (and HWANG Jae-Jeong, and KIM Jung-Eun (1) ;. An agent for the HCARD model in the distributed environment. *Lecture Notes in Computer Science. Computational Intelligence and Security*, 3801:1082–1087, 2005.

[36] D. Daras, D. Bechtsis, and M. Strintzis. A web/wap-based system for remote monitoring patient with data mining support. *Neural Network Applications in Electrical Engineering*, pages 59–64, 2002.

[37] R. Davenport. Etl vs elt. Technical report, Insource IT Consultancy, 2008.

[38] R. Davison, M. Martinsons, and N. Kock. Principles of canonical action research. *Information systems journal*, 14:65–86, 2004.

[39] U. Dayal, M. Castellanos, A. Simitsis, and K. Wilkinson. Data integration flows for business intelligence. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 1–11, 2009.

[40] B. Devlin. Information integration- extending the data warehouse. Technical report, IBM, DB2 Information management software, 2003.

[41] N. Diaz. Bam: Event-driven business intelligence for the real-time enterprise. *DM review*, 3:38–40, 2004.

[42] X. Dong and F. Naumann. Data fusion- resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2, 2009.

[43] Y. Dwivedi, B. Lal, N. Mustafee, and M. Williams. Profiling a decade of information systems frontiers' research. *Information systems Frontiers*, 11(1):87–102, 2009.

[44] W. Eckerson. A business approach to right-time decision making, 2006.

[45] Inc. Electronic Digital Documents. Datacleanser, 2000.

[46] O. Etzion. On real-time, right-time, latency, throughput and other time-oriented measurements, 2007.

[47] E. F. Ewen, C. E. Medsker, and L. E. Dusterhoft. Data warehousing in an integrated health system; building the business case. In *1st ACM international workshop on Data warehousing and OLAP*, pages 47 – 53. Washington, D.C., United States, 1999.

[48] D. Falessi, M. Ali Babar, D. Cantone, and P. Kruchten. Applying empirical software engineering to software architecture: challenges and lessons learned. *Empirical software Engineering*, 15:250–276, 2010.

[49] A. Fard, M. Kahani, R Ghaemi, , and A. Tabatabaee. Multi-agent data fusion architecture for intelligent web information retrieval. *Journal of World Academy of Science, Engineering and Technology*, pages 73–77, 2007.

[50] S. Ferdous, L. Fegaras, and F. Makedon. Applying data warehousing technique in pervasive assistive environment. *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, 2010.

[51] M. Ferguson. Building intelligent agents using business activity monitoring, 2005.

[52] M Figueredo and J. Dias. Mobile telemedicine system for home care and patient monitoring. *26th Annual International Conference of the IEEE EMBS*, pages 3387–3390, 2004.

[53] FIPA. The foundation of intelligent physical agents (FIPA), 2005.

[54] R. Florez-Mendez. Towards a standardization of multi-agent system framework. *ACM Crossroads Magazine*, 5(4):18–24, 1999.

[55] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. *Third International Workshop on Agent Theories, Architectures, and Languages*, pages 1–10, 1996.

[56] M. Golfarelli, S. Rizzi, and L. Cella. Beyond data warehousing: Whats next in business intelligence? *In Proceedings of the 7th ACM international workshop on Data warehousing and OLAP*, pages 1–6, 2004.

[57] N. Gorla. Features to consider in a data warehousing system. *Communications of the ACM*, 46(11):11–115, 2003.

[58] V. Gorodetski, O. Karsayev, and V. Samoilov. Multi-agent data fusion systems: Design and implementation issues. *Proceedings of the 10th International Conference on Telecommunication Systems - Modeling and Analysis*, pages 3–6, 2010.

[59] D. Gregg, U. Kulkarni, and A. Vinz. Understanding the philosophical underpinnings of software engineering research in information systems. *Journal of Information Systems Frontiers*, 3(2):169–183, 2001.

[60] Sh. Gregor and D. Jones. The anatomy of a design theory. *Journal of the Association of Information systems*, 8(5):312–335, 2007.

[61] T. Greiner, W. Dster, F. Pouatcha, R. Von Ammon, H. Brandl, and D. Guschakowski. Business activity monitoring of norisbank taking the example of the application easycredit and the future adoption of complex event processing (CEP). *Proceedings of the 4th international symposium on Principles and practice of programming in Java*, pages 237 – 242, 2006.

[62] J. Guerra and D. Andrews. Creating a real time data warehouse, 2011.

[63] J. Guerrero and J. Garcia, J.and Molina. Multi-agent data fusion architecture proposal for obtaining an integrated navigated solution on uav's. *Proceedings of the 10th International Work-Conference on Artificial Neural Networks*, pages 13–20, 2009.

[64] R. Hackathorn. The bi watch: Real-time to real-value. *DM Review*, 14(1), 2004.

[65] A. Halevy, A. Rajaraman, and J. Ordille. Database integration: The teenage years. *VLDB '06 Proceedings of the 32nd international conference on Very large data bases*, pages 9–16, 2006.

[66] Queensland Goverment Queensland health. *Health information: Disclosure and Access policy*, 2005.

[67] National heart lung and Blood Institute. What is high blood pressure?, 2011.

[68] A. Hevner, S. Chatterjee, S.and Purao, M. Rossi, and M. Sein. Design research in information systems. In *Design Research in Information Systems*, chapter On Integrating Action Research and Design Research, pages 179–194. Springer US, 2010.

[69] A. Hevner, S. March, J. Park, and S Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[70] Y. Hongmei, Z. Jun, J. Yingtao, P. Chenglin, and X. Shouzhong. Selecting critical clinical features for heart diseases diagnosis with real-coded genetic algorithm. *Applied Soft Computing*, 8:1105–1111, 2008.

[71] M. J. Huang, M. Y. Chen, and S. C. Lee. Integrating data mining with case-based reasoning for chronic diseases prognosis and diagnosis. *Expert systems with applications*, 32:856–867, 2007.

[72] P. Huang, H. Lei, and L. Lim. Real time business performance monitoring and analysis using metric network. *Proceedings of the IEEE International Conference on e-Business Engineering*, pages 442–449, 2006.

[73] IBM. Ibm data warehouse manager, 2005.

[74] W. Inmon. The data warehouse and data mining. *Communications of the ACM*, 96(39):49–50, 1996.

[75] W. Inmon. *Building the Data Warehouse (3rd edition)*. Wiley-New York, 2002.

[76] JADE. Java agent development framework (JADE), 2010.

[77] T. Jaorg and S. Dessloch. Near real-time data warehousing using state-of-the-art ETL tools. *The 3rd International Workshop on Enabling Real-Time for Business Intelligence (BIRTE 2009)*, 2009.

[78] M. Javed and A. Nawaz. Data load distribution by semi real time data warehouse. *Proceedings of the 2010 Second International Conference on Computer and Network Technology*, pages 556–560, 2010.

[79] S. Jeffery, G. Alonso, M. Franklin, Wei Hong, and J. Widom. Declarative support for sensor data cleaning. *Lecture Notes in Computer Science In 4th International Conference on Pervasive Computing*, 3968:83–100, 2006.

[80] C. Guangcheng Xi Jianxin, , Yanwei Xing, Jie Wang, and Chenglong Zheng. An unsupervised multi-valued stochastic neural network algorithm to cluster in coronary heart disease data. *Proceedings of the 2007 International Conference on Convergence Information Technology*, pages 640–644, 2007.

[81] T. Jorg and S. Dessloch. Formalizing ETL jobs for incremental loading of data warehouses. *Proceedings der 13. GI-Fachtagung fur Datenbanksysteme in Business, Technologie und Web*, 144(2):327–346, 2009.

[82] G. Kamika, D. Pynadath, and M. Tambe. Monitoring deployed agent teams. *Proceedings of the fifth international conference on Autonomous agents*, pages 308–315, 2001.

[83] J. Kang and H. Han. A business activity monitoring system supporting real-time business perfomance management. *IEEE third 2008 International Conference on Convergence and Hybrid Information Technology.*, 1:473–478, 2008.

[84] B. Kanga, S. Lee, Y. Min, S. Kang, and N. Cho. Real-time process quality control for business activity monitoring. *International Conference on Computational Science and Its Applications*, pages 237–242, 2009.

[85] R. S. Kazemzadeh and K. Sartipi. Incoporating data mining applications into clinical guildelines. In *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, 2006.

[86] A. D. Kemal, D. Laurent, and D. Umeshwar. Towards an architecture for real time decision suport systems:challenges and solutions. *Proceedings of the internartional database engineering and applications symposium IDEAS'01*, pages 303 – 311, 2001.

[87] E. Kendall, P. V. Murali Krishna, C.B. Suresh, and C. Pathak. An application framework for intelligent and mobile agents. *ACM Computing Surveys*, 32(1), 2000.

[88] E. M Kerkri, C. Quantin, F. A. Allaert, Y. Cottin, Ph. Charve, F. Jouanot, and K. Yetongnon. An approach for integrating heterogeneous information sources in a medical data warehouse. *Journal of Medical Systems*, 25(3):2001, 2001.

[89] U. Khot, M. Khot, C. Bajzer, S. Sapp, E. Ohman, S. Brenner, S. Ellis, A. Lincoff, and Topol. E. Prevalence of conventional risk factors in patients with coronary heart disease. *Americal Medical Association*, 7:898–904, 2003.

[90] Ki-Hyeon Kim and Ho-Jin Choi. Design of a clinical knowledge base for heart disease detection. *Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, pages 610–615, 2007.

[91] J. Langseth. Real-time data warehousing: Challenges and solutions, 2004.

[92] C. S. Ledbetter and M. W. Morgan. Toward best practice: Leveraging the electronic patient record as a clinical data warehouse. *Journal of Healthcare Information Management*, 15(2):119–131, 2001.

[93] H. Lee, K. Park, B. Lee, J. Choi, and R. Elmasri. Issues in data fusion for health care monitoring. *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, 3, 2008.

[94] Z. Lin, D. Yang, G. Song, and T. Wang. Dealing with query contention issue in real data warehouse by dynamic multi-level caches. *7th IEEE International Conference on Computer and Information Technology*, pages 122–127, 2007.

[95] Data Ladder Ltd. Data cleaning software, 2009.

[96] J. Maletic and A. Marcus. Data cleansing: Beyond integrity analysis. *Proceedings of the Conference on Information quality*, pages 200–209, 2000.

[97] S. March and G. Smith. Desing and natural science research on information technology. *Decision Support Systems*, 15:251–266, 1995.

[98] F. Marchese. Engineering, science, and design. *In Proceedings of the International Conference on Engineering and Meta-Engineering*, pages 140 –144, 2010.

[99] E. Marcos. Software engineering research versus software development. *ACM SIGSOFT Software egineering notes*, 30(4):1–7, 2005.

[100] R. Mark, A. Mateo, L. Cervantes, Hae-Kwon Yang, and J. Lee. Mobile agents using data mining for diagnosis support in ubiquitous healthcare. *Agent and Multi-Agent Systems: Technologies and Applications*, 4496:795–804, 2007.

[101] V. McBurney. So what is better, ETL or ELT?, 2006.

[102] T Mettler and V. Vimarlund. Understanding business intelligence in the context of health care. *Proceedings of the 13 th International Symposium on Health Information Management Research*, pages 61–69, 2008.

[103] Z. Michalewics, M. Schmidt, M. Michalewics, and C. Chiriac. *Adaptive Business Intelligence*. Springer, 2007.

[104] T. Mikolajczyk. Real-time data integration using change data capture, 2009.

[105] W. Minhong and W. Huaiqing. Intelligent agent supported business process management. *Proceedings of the 38th Annual Hawaii International Conference on System Sciences HICSS'05*, pages 71–82, 2005.

[106] J. Morrison and J. George. Exploring the software engineering component in mis research. *Communications of the ACM*, 38:80–91, 1995.

[107] M. Nascimento, T. zsu, D. Kossmann, R. Miller, J. Blakeley, and K. Schiefer, editors. *Proceedings of the 30th International Conference on Very Large Databases*. Morgan Kaufmann, 2004.

[108] Australia's national agency for health, welfare statistics, and information. Cardiovascular health, 2010.

[109] F. Naumann. *Quality-driven query answering for integrated information systems*, volume 2261. Springer, 2002.

[110] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Computer Society Technical Committee on Data Engineering*, 29(2):21–31, 2006.

[111] S. Negash. Business intelligence. *Communications of the Association for Information Systems*, 13:177–195, 2004.

[112] M. Negoita, R. Howlett, and L. Jain, editors. *Knowledge-Based Intelligent Information and Engineering Systems. Lecture Notes in Computer Science*, volume 3213. Springer Berlin / Heidelberg, 2004.

[113] G. Nelson and J. Wright. Real time decision support: Creating a flexible architecture for real time analytics, 2005.

[114] T. M. Nguyen, J. Schiefer, and A. M. Tjoa. Sense & response service architecture (SARESA): an approach towards a real-time business intelligence solution and its use for a fraud detection application. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 77 – 86. Bremen, Germany, 2005.

[115] J. Nunamaker, M. Chen, and T. Purdin. Systems development in information systems. *Journal of Management Information Systems*, 7:89–106, 1991.

[116] Australian Institute of Health and Welfare (AIHW). *General Record of Incidence of Mortality -Ischaemic heart diseases.* 2007.

[117] P. Offermann, O. Levina, M. Schnherr, and U. Bub. Outline of a design science research process. *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, 2009.

[118] C. Olzak and E. Ziemba. Approach to building and implementing business intelligence systems. *Interdisciplinary journal of information, knowledge and management*, 2:135–148, 2007.

[119] OMG. OMG agent platform special interest group.

[120] Oracle. Oracle 9i - change data capture, 2011.

[121] C. Ordonez, E. Omiecinski, L. De Braal, C. A. Santana, N. Ezquerra, J. A. Taboada, D. Cooke, E. Krawczynska, and E. V. Garcia. Mining constrained association rules to predict heart disease. In *Proceeding of the 2001 IEEE international conference on data mining (ICDM'01)*, 2001.

[122] World Health Organization. Overview - preventing chronic diseases: a vital investment, 2005.

[123] K. Orr. Data quality and systems theory. *Communications of the ACM*, 41(2):66–71, 1998.

[124] L. Padgham and M. Winikoff. *Developing Intelligent Agent Systems*. Wiley, 2004.

[125] UK parliament. Data protection act 1998, 1998.

[126] A. Pawan, Ch.and Lianjun, J. Jun-Jang, and Ch. Shyh-Kwei. Enterprise integration and monitoring solution using active shared space. *Proceedings of the IEEE International Conference on e-Business Engineering*, pages 665 – 672, 2005.

[127] K.V. Pedersen. A framework for a clinical reasoning knowledge warehouse. *Proceedings of the IDEAS Workshop on Medical Information Systems: The Digital Hospital in IEEE database*, 2004.

[128] L. Penserini, A. Perini, A. Susi, and J. Mylopoulos. High variability design for software agents: Extending tropos. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4), 2007.

[129] Kemal Polat, Salih Gunes, and Sulayman Tosun. Diagnosis of heart disease using artificial inmmune recognition system and fuzzy weighted pre-processing. *Patter recognition*, 39:2186–2193, 2006.

[130] PostgreSql. C library, 2010.

[131] PostgreSql. Notify rule, 2011.

[132] N. Prata, J. Akokab, and I. Comyn-Wattiau. A UML-based data warehouse design method. *Journal of Decision Support Systems*, 42(3):1449–1473, 2006.

[133] N. Raden. Exploring the business imperative of real-time analytics. *HIRED BRAINS, INC. Implementing Business Analytics.*, 2010.

[134] N. Raden and R. Kimball. Real time: Get real, part ii, 2003.

[135] E. Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, pages 3–13, 2000.

[136] Arvind Arasu-Brian Babcock Shivnath Babu Mayur Datar Gurmeet Manku Chris Olston Justin Rosenstein Rohit Varma Rajeev Motwani, Jennifer Widom. Query processing, approximation, and resource management in a data stream management system, 2003.

[137] M. X. Ribeiro, A. J. M. Traina, C. Traina, N. A. Rosa, and P. M. A Marques. How to improve medical image diagnosis through association rules: The IDEA method. *21st International Conference on Data Engineering IEEE International symposium on computer-based medical systems*, pages 266–271, 2008.

[138] R. Richey and J. Klein. *Design and Development Research: Methods, Strategies, and Issues*. Lawrence Erlbaum Associates, Inc., 2007.

[139] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, 2/E*. Prentice Hall, 2003.

[140] Microsoft S.A. SQL SERVER 2012 - change data capture, 2012.

[141] T.R. Sahama and P. R. Croll. A datawarehouse architecture for clinical data warehousing. In *Proceedings of the fifth Australasian symposium on ACSW frontiers*, volume 68, pages 227 – 232. Ballarat, Australia, 2007.

[142] A. M. Salem, M. Roushdy, and R. A. Hodjod. A case based expert system for supporting diagnosis of heart disease. 2003.

[143] R. Santos and J. Bernardino. Real time data warehouse loading methodology. *Proceedings of the 2008 international symposium on Database engineering and applications*, pages 49–58, 2008.

[144] J. Schiefer, B. List, and R. M. Bruckner. Process data store: A real-time data store for monitoring business processes. *Lecture notes in computer science, Springer Berlin / Heidelberg*, 2736:760–770, 2003.

[145] A. Sen and A. P. Sinha. A comparison of data warehousing methodologies. *Communications of the ACM*, 48(3):79–84, 2005.

[146] S. Shah, S. Dharmarajan, and K. Ramamritham. An efficient and resilient approach to filtering and disseminating streaming data. *Proceedings of the 29th international conference on Very large data bases*, 29:57 – 68, 2003.

[147] M. Shaw. The coming-of-age of software architecture research. *ICSE '01 Proceedings of the 23rd International Conference on Software Engineering*, pages 656–665, 2001.

[148] M. Shawa, Ch. Subramaniama, G. Tana, and M. Welgeb. Knowledge management and data mining for marketing. *Decision Support Systems*, 31(1):127137, 2001.

[149] G. Shelly, T. Cashman, and H. Rosenblatt. *Systems Analysis and Design*, chapter 11, page 731. Course Technology, 2009.

[150] A. Simitsis, P. Vassiliadis, and T. Sellis. State-space optimization of ETL workflows. *IEEE Transactions on knowledge and data engineering*, 17(10):1404–1419, 2005.

[151] A. Simitsis, P. Vassiliadis, and T.K. Sellis. Optimizing ETL processes in data warehouses. *21st International Conference on Data Engineering*, pages 564–575, 2005.

[152] A. Simitsis, K. Wilkinson, M. Castellanos, and U. Dayal. Qox-driven ETL design: reducing the cost of ETL consulting engagements. *Proceedings of the 35th SIGMOD international conference on Management of data*, 2009.

[153] A. Singhal and D. Seborg. Matching patterns from historical data using pca and distance similarity factors. *Proceedings of the American Control Conference*, pages 1759–1764, 2001.

[154] D. Snoddy, J. Spyker, M. Rupik, M. Jory, and K. Kobylinski. Change data capture:what is it and how it impacts solutions architecture. *Proceedings of the 2099 conference for advanced studies on collaborative research*, pages 297–298, 2009.

[155] M. Soffner, M. Siegmund, M. Pukall, and V. Otto-von Guericke. Towards real-time data integration and analysis for embedded devices. *Proceedings of the GI-Workshop on Foundations of Databases*, pages 51–55, 2009.

[156] T. Spil, R Stegwee, and Ch. Teitink. Business intelligence in healthcare organization. *35th Annual Hawaii internation conference on system sciences*, page 142b, 2002.

[157] P. Srinivas Sajja. Multi-agent system for knowledge-based access to distributed databases. *Interdisciplinary Journal of Information, Knowledge, and Management*, 3:1–9, 2008.

[158] S Srinivasan, V Krishna, and S. Holmes. Web-log-driven business activity monitoring. *IEEE Computer Society*, 38(3):61 – 68, 2005.

[159] J. Sutherland and W. Van den Heuvel. Towards an intelligent hospital environment: Adaptive workflow in the OR of the future. *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, 5:100– 110, 2006.

[160] W. J. Sutherland, J.and Van den Heuvel. Clinical process and data integration and evolution. *40th Annual Hawaii International Conference on System Sciences in IEEE database*, 2007.

[161] N.B. Szirbik and T. Pelletier, C. andChaussalet. Six methodological steps to build medical data warehouses for research. *International Journal of Medical Informatics*, 75(9):683–691, 2006.

[162] D. Tank, A. Ganatra, Y. Kosta, and C. Bhensdadia. Speeding ETL processing in data warehouses usng high-perfomance joins for changed data capture. *2010 International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom)*, pages 365–368, 2010.

[163] R. Taylor. Concurrency in the data warehouse. *VLDB 2010 , 36th International Conference on Very Large Data Bases*, pages 724–727, 2000.

[164] S. Terr. Real-time data warehousing 101, 2004.

[165] V. Thion-Goasdou, S. ; Nugier, D. Duquennoy, and B. Laboisse. An evaluation framework for data quality tools. *International Conference for Information Quality (ICIQ)*, 2007.

[166] CH. Tseng and P. Gmytrasiewicz. Real time decision support system for portfolio management. *Hawaii International Conference on System Sciences*, 3:79, 2002.

[167] V. Vaishnavi and W. Kuechler. Design research in information systems. *Journal On The Theory Of Ordered Sets And Its Applications*, 48(2):133–140, 2011.

[168] C. Van Aart, R. Pels, G. Caire, and F. Bergenti. Creating and using ontologies in agent communication. *Creating and Using Ontologies in Agent Communication. In Proceedings of 1st International Conference on Autonomous Agents & Multiagents.Second International Workshop on Ontologies in Agent Systems*, 2002.

[169] J. Van Bemmel, P. Dockhorn, and I. Widya. Paradigm: Event-driven computing. Technical report, Lucent Technologies, 2004.

[170] P. Vassiliadis and A. Simitsis. *Near Real Time ETL*, volume 3. Springer US, 2009.

[171] N. Wenger. Clinical characteristics of coronary heart disease in women: emphasis on gender differences. *Cardiovascular Research*, 53(3):558–567, 2002.

[172] N. Wickramasinghe, C. Guttmann, and J. Schaffer. Designing intelligent healthcare operations. *Lecture Notes in Computer Science. Collaborative Agents - Research and Development*, 6066:152–162, 2011.

[173] R. Wieringa. Design science as nested problem solving. *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, 2009.

[174] R. Wieringa and H. Heerkens. Desin science, engineering science and requirements engineering. *16th international requirements engineering conference*, pages 310–313, 2008.

[175] R. Wieringa, N. Maiden, and N. Mead. Requirements engineering paper classification and evaluation criteria: a proposal discussion. *Requierements Engineering*, 11:102–107, 2006.

[176] P. Wilson, R. D' Agostino, D. Levy, A. Belanger, H. Silbershatz, and W. Kannel. Prediction of coronary heart disease using risk factor categories. *Journal of the American Heart Association*, 97:1837–1847, 1998.

[177] B. Wu and K. Dube. Plan: a framework and specification language with an event-condition- action (eca) mechanism for clinical test request protocols. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pages 140–150, 2001.

[178] Yanwei Xing, Jie Wang, Zhihong Zhao, and Yonghong Gao. Combination data mining methods with new medical data to predicting outcome of coronary heart disease. *Proceedings of the 2007 International Conference on Convergence Information Technology*, pages 868–872, 2007.

[179] Y. Yan, W. Li, and J. Xu. Information value-driven near real-time decision support systems. *29th IEEE international conference on Distributed Computing Systems. ICDCS '09*, pages 571–578, 2009.

[180] Y. Yin and D. Papadias. Just-in-time processing of continuous queries. *IEEE 24th International Conference on Data Engineering*, pages 1150–1159, 2008.

[181] Y. Ying and L. Wen-Sian. Correlation aware synchronization for near real time decision support systems. *Proceedings of the 13th International Conference on Extending Database Technology*, pages 39–50, 2010.

[182] Ch. Zhang and S. Zhang. Association rule mining: models and algorithms. In G. Goos, J. Hartmanis, and G. Van Leeuwen, editors, *Lecture Notes in Artificial Intelligence*, page 146, 2002.

[183] Y. Zhu, L. An, and L. Shuangxi. Data updating and query in real-time data warehouse system. *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, 5:1295–1297, 2008.

[184] J. Zuters. Near real-time data warehousing with multi-stage trickle & flip. In *Perspectives in Business Informatics Research*, volume 90 of *Lecture Notes in Business Information Processing*, pages 73–82, 2011.